Supporting User Appropriation of Public Displays

Sarah Elizabeth Clinch
M.Sc (Lancaster, 2008)
B.Sc (Lancaster, 2006)

School of Computing and Communications

Lancaster University

A thesis submitted for the degree of

Doctor of Philosophy

December 2013

Declaration and Acknowledgements

This thesis has been written by myself. The work reported in this thesis has not been previously submitted for a degree in this or any other form. Some of the work reported in this thesis has previously appeared in a different form in published papers and articles [CDFC13, CDFE11a, CDKS12, CHF+12, CKDL12a, CKDL12b, Cli13, DCAss, DFCS10, DFN+09a, DLC+14, HPL+13, KCDL12] on which I was a contributing author.

The software infrastructure described in this thesis was developed as part of the PD-NET project and was supported by European Union Seventh Framework Programme (FP7/2007–2013), grant number 244011.

The following paragraphs indicate portions of the reported work produced as a result of collaborations with colleagues at Lancaster University and other institutions. A clear summary of this author's contributions to these collaborative works is given in Table 3.1 (page 95) for the initial probes and Table 4.1 (page 150) for the final architecture.

Understanding Appropriation: Probes

The Bluetooth Device Name system that was used for the studies forming the probe described in Section 3.3.1 was implemented by Peter Newman and Oliver Storz. Performance measurements for this probe were produced by Oliver Storz. My contribution in this probe was a focussed analysis of the system in the context of display appropriation and the execution of a study that explored potential use cases.

The cloud and cloudlet performance data analyses summarised in Section 3.3.2.3 and the statistics describing display owner usage practices of the e-Channel system summarised in Section 3.3.3.2 were produced as a joint work with Adrian Friday.

The viewer interviews reported in Section 3.3.3 were designed by, and issued to participants, in collaboration with Kevin Smith.

Design, Implementation and Evaluation

A number of individuals contributed to the implementation of Yarely reported in Section 5.2. Mac OS and Linux GNOME renderers were produced by Graham Clinch. The Linux Raspian adaptations were developed by Adrian Friday. The Yarely scheduler was the result of joint work with Adrian Friday and Bholanathsingh Surajbali.

The implementation of Mercury reported in Section 5.4 was a joint work with the majority produced by Mateusz Mikusz and Miriam Greis. This implementation was based on an initial idea and subsequent refinements produced by myself and published in Clinch et al. [CDKS12] and Davies et al. [DFCS10].

The original design concept for Tacita (as presented in Section 4.5) was developed as joint work with Nigel Davies, Marc Langheinrich, Ivan Elhart, Adrian Friday, Thomas Kubitza and Bholanathsingh Surajbali. The Tacita mobile clients described in Section 5.5.2.1 were implemented by Thomas Kubitza and Christopher Winstanley. The Tacita PHP map provider described in Section 5.5.2.3 was implemented by Thomas Kubitza who also implemented the *PD News* and *ubiVM* applications described in Section 5.5.2.4. *newsBounce* (Section 5.5.2.4) was the result of joint work with Mateusz Mikusz, Christopher Winstanley and Mike Harding. Within the Tacita implementation, my contributions were focussed on personalised applications: *World Clock*, *PD Weather*, *native-VNC* and *newsBounce*.

The quantitative measurements evaluating the use of different proximity methods (presented in Section 6.2.3) was completed by Thomas

Kubitza and Bholanathsingh Surajbali.

Acknowledgements

I'd like to thank my supervisor Prof. Nigel Davies for his support and encouragement throughout my Ph.D. with regards to my research, but also more generally in providing opportunities to facilitate collaborations and develop my future career. Thanks are also due to Dr. Adrian Friday who provided valuable feedback and expert advice throughout the Ph.D. process and to Prof. M. Satyanarayanan who made a great deal of time for me during (and following) my three month period as a visiting scholar at Carnegie Mellon University.

Thank you to Albrecht Schmidt, Marc Langheinrich, Rui José and other members of the PD-NET consortium who provided numerous encouragements during the duration of the project as well as offering advice and feedback about the work. Working on PD-NET was a very rewarding experience and I feel privileged to have been involved in such a successful project. Special thanks are extended to Mateusz Mikusz and Miriam Greis for their work on the implementation of Mercury and to Thomas Kubitza for his development of the Tacita components.

My thanks also go to the various colleagues with whom I have shared office space (and to John Hardy with whom I share only a supervisor) who have helped to create a pleasant and productive work environment, and who have on occasions been known to down-tools and help meet a tight deadline or resolve a seemingly impossible LaTeX or Python bug.

I owe a great debt to my father for introducing me to programming and generally fostering an interest in maths and logic, and my parents for their encouragement to pursue education to this level. Finally, I thank Graham Clinch for his patience, technical expertise, and frequent encouragement at all stages in my research, without him this would undoubtably have been a much harder achievement.

Abstract

Supporting User Appropriation of Public Displays

Sarah Elizabeth Clinch

M.Sc (Lancaster, 2008)
B.Sc (Lancaster, 2006)
Submitted for the degree of Doctor of Philosophy.

December 2013

Despite their prevalence, public engagement with pervasive public displays is typically very low. One method for increasing the relevance of displayed content (and therefore hopefully improving engagement) is to allow the viewer themselves to affect the content shown on displays they encounter – for example, personalising an existing news feed or invoking a specific application on a display of their choosing. We describe this process as *viewer appropriation* of public displays.

This thesis aims to provide the foundations for appropriation support in future 'open' pervasive display networks. Our architecture combines three components: Yarely, a scheduler and media player; Tacita, a system for allowing users to make privacy-preserving appropriation requests, and Mercury, an application store for distributing content. Interface points between components support integration with third-party systems; a prime example is the provision of Content Descriptor Sets (CDSs) to describe the media items and constraints that determine what is played at each display.

Our evaluation of the architecture is both quantitive and qualitative and includes a mixture of user studies, surveys, focus groups, performance measurements and reflections. Overall we show that it is feasible to construct a robust open pervasive display network that supports viewer appropriation. In particular, we show that Yarely's thick-client approach enables the development of a signage system that provides continuous operation even in periods of network disconnection yet is able to respond to viewer appropriation requests. Furthermore, we show that CDSs can be used as an effective means of information exchange in an open architecture. Performance measures indicate that demanding personalisation scenarios can be satisfied, and our qualitative work indicates that both display owners and viewers are positive about the introduction of appropriation into future pervasive display systems.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

1.1 Overview

Pervasive public displays have long been a rich vein of academic research. As hardware costs have decreased, the prevalence of digital displays in public spaces has grown considerably; displays of varying sizes, shapes and forms are now commonly seen in everyday spaces. Installations vary in form factor, and purpose: paper timetables have given way to digital information displays and digital advertisements are increasingly ubiquitous.

Digital displays offer the potential to enhance our public spaces. Digital signage improves over traditional notices by facilitating frequent, timely updates, increasing accuracy and enabling provision of highly-dynamic multimedia information that would not otherwise have been made available. Such displays also offer new possibilities for improving the aesthetics of a space by allowing the presentation of digital artwork, video or other media. Furthermore, the introduction of personalised, interactive content on public displays has the potential to promote viewer engagement with the space and to encourage social interaction within the space.

The creation and distribution of engaging content is key to realising the potential offered by the installation of digital displays in public spaces. Despite their prevalence, digital displays currently fail to hold viewers' attention, instead viewers have become skilled at ignoring them [MWE+09]. Typical commercial display deployments are dominated by generalised advertising content which is broadcast regardless of the current audience and rarely tailored to the individuals viewing the display. Furthermore, content on such systems is tightly-controlled; if an individual has an idea for content that would be appropriate for a public display it would, in most cases, be extremely challenging for them to be able to negotiate the technical and economic hurdles to placing their content on a display. By contrast, researchers predict a movement to 'open' displays [DLJS12] arranged into large-scale networks [SKM09b, SKM09a] in which content from multiple sources can be easily integrated.

Moving away from generalised content and improving the relevance of displayed content to passing viewers has the potential to increase engagement. One method for ensuring the relevance of displayed content is to allow the viewer themselves to affect the content shown on the displays they encounter – for example, requesting that content shown on nearby displays reflects their interest in a particular sport or hobby or even invoking a specific application on a display of their choosing. We describe this process as viewer 'appropriation' of public displays.

In this thesis, we consider mechanisms for supporting viewer appropriation of public displays. We explore aspects of the design space and provide an architecture and implementation to support appropriation in an open display network.

1.2 Towards Open Public Display Systems

Early deployments of digital displays into shared spaces typically consisted of one or two screens deployed for the purpose of providing a shared noticeboard, building communities or for joining spaces with video links (see Section 2.2). As hardware costs have reduced and a commercial market for digital signage products has emerged, such deployments have grown in size and frequency. Digital displays are now a common presence in a variety of spaces.

Whilst the overall frequency of public displays has increased, the scale of a typical installation remains relatively small – an organisational digital signage infrastructure might consist of tens of displays – although the emergence of digital advertising networks is creating larger deployments (for example, DSN [Dig12] has over a thousand displays across India). Moreover, each deployment operates under a single management domain and assumes traditional, static, broadcast models for content playback; management users create playlists and linear schedules (for example using tools such as Sony Ziris [Son14], Scala [Sca14] and signagelive [Rem12]).

A move towards large networks of digital public displays was predicted by Strohbach et al. [SKM09b, SKM09a]. They suggested that such systems would require middleware to enable them to act as "an intelligent collection of heterogeneous geo-localised displays that... adapt their content to... users, ...location and context" [SKM09b, p2]. Features of such middleware would include high-level programming abstractions to allow content distribution to the whole network (rather than addressing specific displays individually), viewer tracking, privacy protection, and open APIs.

Davies et al. [DLJS12] suggest a model for future public display systems in which future deployments would take the form of large-scale 'open' networks. Such systems contrast with the current, 'closed', display- and content-management processes by supporting content and applications from a wide variety of sources. Future content sources might include businesses of varying sizes, display viewers and other individuals, and even the display environment itself.

Proponents of an open approach for future display systems suggest that introducing openness has the potential to transform the medium, creating a valuable communications channel and reversing the trend for 'Display Blindness' [MWE+09], in which passers-by fail to attend to digital displays. In the following two scenarios we illustrate how such future display networks might work and their benefits over current deployments.

1.2.1 Scenario 1: Missing Child

The following scenario is drawn from research literature and first appears in Davies et al. [DLJS12]:

Sue realises she has lost her daughter, Millie, in the local shopping centre. She immediately calls the police who ask her to send them a recent photo through her mobile phone. Within seconds all of the displays in the shopping centre are *showing a photograph of Millie together with a number to call. As the minutes pass the photographs spread over an increasing area – reflecting how far away Millie might now be. Within five minutes Millie and her mum are reunited; Millie was found by a shopper who recognised her photo from the screen.*

In this scenario we see how moving away from independent display systems, each under a different management domain, offers the potential for Millie's photograph to easily spread over an increasing geographic area. We see a change away from small, isolated sets of displays towards something that can be viewed as a single, large, display network. A similar transformation could be seen in computer networking following the emergence of the Internet: prior to this computers were isolated, or networked in small clusters to facilitate resource sharing. The transformation that occurred as mechanisms emerged to enable communication between computers on different networks, creating a large-scale open platform, has had profound implications on the way we think about and use computers.

The missing child scenario also illustrates a more open scheduling approach than exists in current linear models. As time passes the missing child alert appears on a new set of displays; each display schedule is open to interruption. One can easily see how this transfers to other forms of time-critical alert, perhaps in an emergency situation, but it also creates the potential for scheduling based on environmental triggers (for example, the presence of a viewer or a sudden local climate change). In this scenario, the content itself is unknown until the triggering event occurs, offering potential for new and interesting forms of content – perhaps generated by local viewers or the environment.

1.2.2 Scenario 2: World Clock

Jack's girlfriend is on a gap year in Australia. As he goes about his day, he periodically calculates the time difference and speculates as to what she might be doing. On his way to work one Monday he notices a 'World Clock' application running on the display in the window of the travel agent's opposite his workplace – it is currently showing the time in San Francisco and Yerevan. He pulls out his mobile phone and opens his Display-Apps store window. He enters 'World Clock' into the search box and selects the application from the results. Running *the application on his phone for the first time he creates a new location of interest for Perth, Australia. Next time he passes the display he is pleased to see the World Clock application appears on the display and he can use the clock for Perth to help him imagine what his girlfriend might be up to. Sometimes the clock for Perth is the only one shown in the application whilst sometimes it appears alongside other locations – perhaps he will take his girlfriend on holiday to one of these new locations he is discovering when she gets back.*

In this scenario we see a demonstration of viewer-driver personalised content. A simple application accepts parameters to customise the content shown at the display. Interactivity is achieved through communication from a viewer's personal mobile device (in this case, a smartphone). Whilst this application accepts only text input (used to specify locations of interest), other applications might require more fine-grained and continuous interaction (as described in Section 3.2). Similarly, some applications may provide output to a viewers's mobile device, allowing them to take away a piece of information or token for future use. Such applications are considerably more complex than current display content and have the potential to significantly improve the value for viewers by responding to current needs or interests and actively engaging them through their interactivity. Finally, the presence of multiple clocks on the screen shows how multi-user interaction is supported for this application.

1.2.3 Personalised, Viewer-driven Content

In this thesis we focus on one potential content source for such future open display networks, that of content sourced from the viewer themselves. We consider the case in which individual viewers can affect the content shown on the displays they encounter. We describe such content as *personalised* or *viewer-driven* content, and the process of placing that content on the screen as *appropriation*.

In systems supporting viewer appropriation a viewer may, for example, request that content shown on nearby displays reflects their particular sports or news interests, or may even invoke a specific application on a display of their choosing. This openness to viewer-driven selection of content is in direct contrast to traditional broadcast models of content distribution in which advertisers and display owners select the content to be shown.

1.3 Contributions of this Thesis

This thesis contributes to existing public display research by providing a detailed exploration of viewer appropriation of pervasive public displays and providing an architecture that supports viewer appropriation in current and future display networks.

The key contributions of this work are as follows:

C1 *A detailed survey of research into pervasive display systems.* Our primary contribution in this space is a focussed exploration of personalisation and appropriation support within previous pervasive display research. The review indicates that whilst some provision for viewer personalisation has been included in a small set of works, it is not typically made available in today's digital display deployments.

C2 *Understanding of user attitudes to appropriation.* We explore both display owner and viewer attitudes towards display appropriation, demonstrating that display owners are willing to allow appropriation and identifying factors that influence the trust that viewers place in unknown content shown on a display infrastructure.

C3 *Detailed usage models and requirements for an appropriation architecture.* We provide a set of requirements for system support for appropriation in open display networks based on reflections around scenarios and experimental probes. In doing so, we identify a set of distinct models for display appropriation including *walk-by personalisation*, *longitudinal personalisation* and *active personalisation.*

C4 *A unified pervasive display architecture with explicit support for display appropriation.* We present the design, implementation and evaluation of an architecture for supporting viewer appropriation and personalisation of future public display networks. In particular, we provide:

C4.1 *A scheduler and media player for open display networks.* We describe a component-based scheduling and playback system that is easily extensible for new forms of content (e.g. complex interactive applications) and scheduling constraints (e.g. the presence of viewers, environmental change), but that also maintains interoperability with an existing signage solution showing that such a platform can continue to support traditional linear scheduling when necessary.

C4.2 *The design of an application store for distributing content within open pervasive display networks.* We present an application store to support content distribution in open display networks and encourage a separation of roles between those who create content and applications (e.g. advertisers, designers and developers) and those who consume it (i.e. display owners). The application store is designed to support a range of business models and exports display schedules in the Content Descriptor Set document format.

C4.3 *A mechanism for describing content availability within open pervasive display networks.* We define a format for exchanging documents that describe the content and constraints available for playback at a display node. These Content Descriptor Set documents are agnostic of any network protocols used to exchange them and are designed to support many-to-many relationships between display nodes and content sources.

C4.4 *Support for viewer display appropriation requests.* We provide mechanisms for allowing viewers to use their personal mobile devices to make appropriation requests of pervasive display infrastructure.

C5 *An evaluation of the integrated software architecture* that demonstrates that our unified architecture meets the requirements presented as part of C3.

1.4 Context and Testbed: e-Campus

Much of the work described in early sections of this thesis uses the e-Campus system [FDE12, SFD06a, SFD+06b, Sto08] as an experimental testbed. In the this section we provide a summary of the e-Campus system and its usage on the Lancaster University campus.

1.4.1 Overview

e-Campus is a software platform and a networked deployment of public displays on the campus of Lancaster University in the UK. The system is designed to serve a dual role: as a day-to-day digital signage solution and as an infrastructure and testbed for research and artist installations.

For the majority of the time, e-Campus displays serve staff- and student-authored content – typically posters and slideshows composed of text and images – as a mechanism for improving the experience of staff, students and visitors on campus. Such content may be shown University-wide (e.g. in the case of news released by the University Press Office) or may be location-specific (e.g. in the case of changes to the schedule of lectures in a nearby space).

In addition to the above, the displays are capable of supporting audio, video and interactive applications, and such features are used by local researchers and artists for whom e-Campus serves as a useful testbed.

1.4.2 Computational Model and APIs

The e-Campus system was designed to support the creation of new applications and featured a set of programming interfaces.

The e-Campus computational model and accompanying low- and high-level APIs allow lifecycle management and presentation of content on displays within the network. The model is comprised of six entities [Figure 1.1]:

- A **`Display`** abstracts over underlying hardware to provide an observable outlet for content. A `Display` provides operations (exposed through a low-level API) that allow the creation, transition and termination of

`Applications` (see below) and uses priority levels to arbitrate between conflicting content items.

- An **`Application`** renders a content item onto a `Display`.
- A **`Handler`** is an optional policy component that provides some additional functionality (e.g. resource locking for shared hardware, audit trail provision).
- A **`Scheduler`** uses the `Application` creation, termination and transition operations provided by the low-level API to control the life-cycle of one or more `Applications` on one or more `Displays`. Multiple `Schedulers` may operate on one `Display` simultaneously.
- A **`Context Source`** may be accessed by `Applications` and `Schedulers` to provide information about the operating environment.
- An **`Interaction Device`** can provide interaction events to `Applications` and `Schedulers`.

The low-level scheduling API provides a limited number of operations for manipulation of `Display` and `Application` objects: `CreateApplication()` and `TerminateApplication()`, for the instantiation and termination of `Application` processes respectively; `ChangeState()`, to instigate the prefetching of `Application` content; and `Transition()`, to toggle the visibility of `Application` output on a `Display`. API operations may be grouped into transactional blocks, providing support for complex, multi-display scheduling.

For those with sufficient programming expertise, the low-level scheduling API allows the development of custom schedulers, giving precise control over scheduling constraints. However, in the majority of cases, the superset of scheduling constraints are based on relatively few requirements: restriction according to date, time, location, priority and synchronisation. For this reason, e-Campus provides a "high-level scheduling service", a system-wide scheduler with support for these common scheduling constraints. The high-level scheduling service provides the following entities:

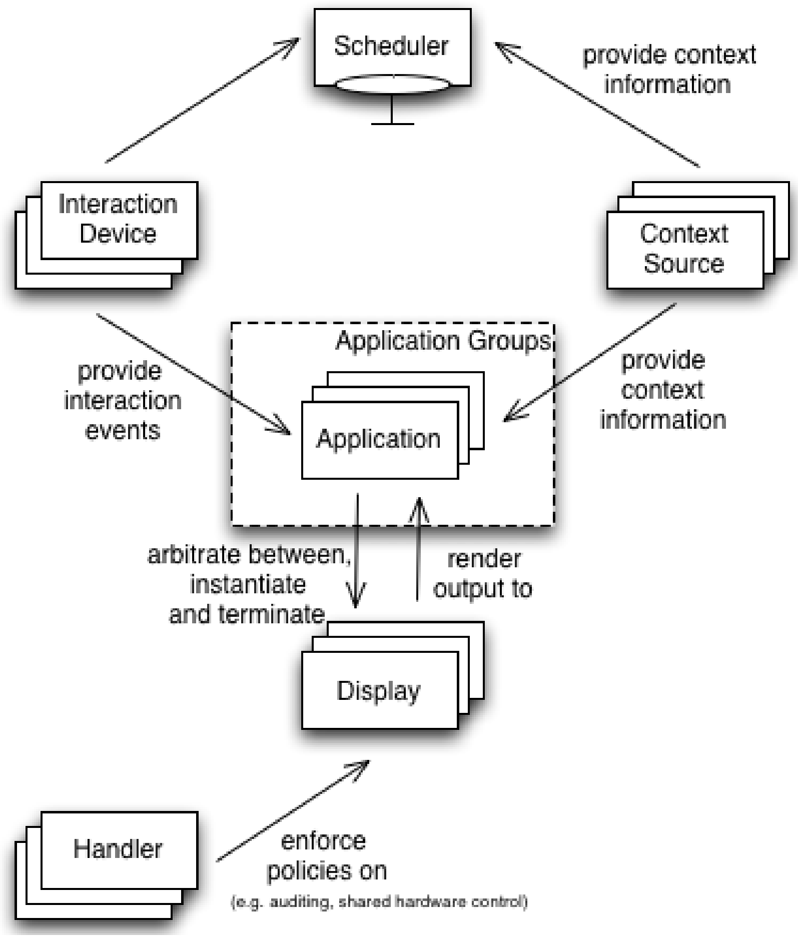

Scheduler
provide context information
Interaction Device
Context Source
Application Groups
provide interaction events
Application
provide context information
arbitrate between, instantiate and terminate
render output to
Display
Handler
enforce policies on
(e.g. auditing, shared hardware control)

Figure 1.1: An Overview of the e-Campus Computational Model.

- **`Playlists`** represent sets of content items and internal constraints such as ordering and within-playlist synchronisation.
- `Playlist Requests` (**`Requests`**) specify external constraints such as date, time, priority and between-playlist synchronisation.

A high-level API is built on top of this scheduling service to allow management of `Playlist` and `Request` objects. The following operations are made available in this API: `CreatePlaylist` (`CreateRequest`), `UpdatePlaylist` (`UpdateRequest`), `DeletePlaylist` (`DeleteRequest`), for the creation, management and deletion of `Playlists` (or `Requests`) respectively, `RetrievePlaylist` (`RetrieveRequest`) and `ListPlaylists` (`ListRequests`) for the retrieval of stored `Playlists` (or `Requests`) and `ClonePlaylist` to allow duplication of existing `Playlists`. The high-level API operations may be invoked through use of HTTP GET requests, either programmatically or through a traditional web browser, removing the need for programming expertise.

Once the high-level scheduling service has selected a `Playlist`, the low-level API is used to show the associated content on the target display(s).

1.4.3 Implementation and Deployment

The low-level software model is implemented in Python; entities within the model communicate using Elvin notifications and subscriptions [SA97]. The low-level API is implemented in Python and provided as a Python-based API. The high-level scheduling service and API are also implemented in Python, but this API is made HTTP accessible through a CGI script on an Apache web server.

The low-level software model processes execute on Mac Mini computers within the e-Campus networked deployment. The network, established in 2005 has gradually expanded to its current state of approximately thirty large LCDs and one projected display, plus around forty associated digital doorplates. The displays are deployed in a variety of locations, including foyers of lecture theatres and other academic buildings, shared spaces in residential buildings, bars and social spaces.

Each display (LCDs and projections) is driven by a Mac Mini computer running Mac OS X. At the beginning of the work described in this thesis, the e-

Campus infrastructure consisted of approximately twenty-five Mac Mini computers (typical specification 1.83GHz Intel Core Duo, 2GB RAM) running Mac OS X 10.4, each accompanied by a 40" LCD (with one exception, a projected display shown in Figure 1.2b); a number of sites also included loudspeakers affixed to (e.g. Figure 1.2c), or adjacent to (e.g. Figure 1.2a), the display. During the work described in this thesis, the deployment was expanded and newer machines and operating systems added (further details in Section 5.2).

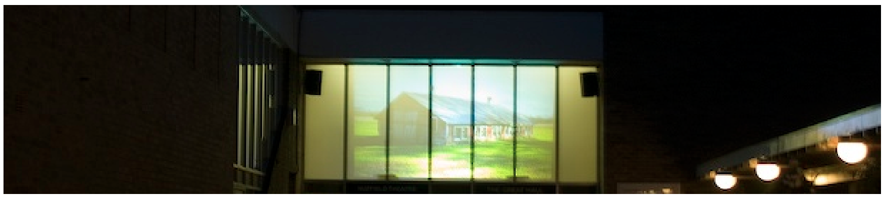
1.4.4 e-Channels

The e-Channel System is the main mechanism for content submission to the day-to-day digital signage player executing on the e-Campus deployment. The main premise of the system is a distinction between two operating roles: content provision and display ownership (support is also given for users who wish to adopt both roles – for example, a display owner controlling displays in a social space who also acts as content provider, creating content to highlight upcoming events in that space).

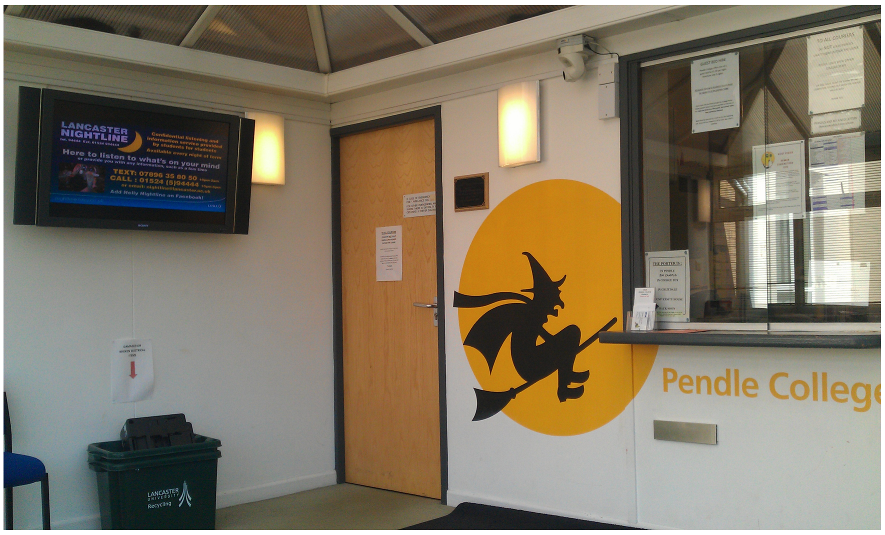
Content Providers Content providers generate content in a range of media formats (images, videos, web pages, media streams) and organise their content in logical containers called "Channels". A content provider has full ownership of their Channels: at any time they may edit the content within their Channel, schedule the times at which the Channel is made available and suspend or delete the Channel. Content providers have no affiliation with displays in the infrastructure and have no control on where or when content is displayed (although some content providers may acquire this control by acting in both a content provision *and* a display ownership role). For a content provider's content to be displayed at a display, they must share the containing Channel and have display owners subscribe to that Channel.

Display Owners Display owners are linked to one or more physical displays in our deployment, typically, those in a 'local' physical space. For example, our College residence officers are often the display owners for displays located within their College's premises and departmental administrators may 'own'

(a) Great Hall LCD



(b) Great Hall projected display



(c) Pendle College LCD

Figure 1.2: In-Situ Photographs of Deployed e-Campus Displays

displays within a department – display ownership often coincides with a general responsibility for the surrounding physical space.

A display owner controls the content on their displays through Channel subscriptions; at any time they can alter the subscriptions for their display and decide when the display should be scheduled to power on or off.

The e-Channel system provides two user interfaces: a web interface, for both display owners and content providers, that allows configuration of display availability (display owners) and channel subscriptions (display owners) and the creation and modification of channels (content providers); and a file share that allows content providers to add and remove content items to their Channels. The web-based portions of the e-Channel system are built using PHP – the file store is scanned by a Python process which stores an e-Campus `Playlist` in a MySQL database for playback back by the High Level Scheduler.

e-Channels is an important foundation for the early work presented in this thesis that explores display owner and viewer attitudes to unknown content in a digital signage platform (Section 3.3.3), and is later integrated as a legacy system that can supply content to the architecture described in this thesis (Section 6.3.2.2).

1.5 Roadmap

The remainder of this thesis is structured as follows:

Chapter 2 provides the background and motivation for the work; this includes a survey of significant public display/signage systems and of work exploring audience responses to such systems, highlighting recent research that suggests typical passers-by do not attend to the digital displays in their environment. The viewer appropriation work presented in this thesis represents one potential future content source that might increase the value of public displays for their viewers and the chapter therefore closes with a review of existing work in this space, i.e. projects that allow user personalisation or appropriation.

Chapter 3 identifies a set of requirements for systems supporting viewer appropriation/personalisation of pervasive public displays. It begins by providing an extensive set of motivating scenarios covering a wide range of usage patterns for appropriation of displays in public spaces. It continues by describing work we have conducted in the form of probes into this space - including both highly-constrained display personalisation using Bluetooth device names, and highly-flexible display appropriation through use of virtualisation and VNC technologies. The chapter explores the role of trust in display networks, particularly those open to appropriation by passing users, and provides an example of the e-Campus system, in which it is demonstrated that display owners are more willing to open up their displays to content from other sources than one might initially expect. The chapter concludes with an analysis of the requirements elicited through the probes.

Chapter 4 describes the design of an architecture for appropriation of pervasive display networks. The chapter includes an overview of the architecture and its interaction with existing (and future) systems. The chapter also includes a detailed description of the core components, namely a display node client for the scheduling and playback of a range of content types, an application store for public display applications, and a smartphone client to support display appropriation requests from viewers. The chapter also describes a medium for exchanging content items, constraints and schedules between components within the architecture.

Chapter 5 describes the build and deployment of our architecture, providing implementation detail for Yarely, a client-side scheduler and media player; Content Descriptor Sets, a document format for representing content items and constraints; Mercury, an application store for application selection; and Tacita, a mechanism to support viewer-generated display appropriation requests. We describe the deployment of the architecture, as an extension of the existing e-Campus platform.

Chapter 6 provides an evaluation of the delivered architecture. It includes quantitative data based on usage patterns and technical measurements (e.g. statistics on application use), plus qualitative data gathered from different stakeholders (e.g. insights into the application development/deployment process, data from user studies).

The final chapter provides a summary of the contents and contributions of the thesis. The chapter begins by identifying the key contributions of the work and continues by identifying areas for future work in the space. The chapter concludes with a set of closing remarks.

1.6 Summary

This thesis examines the topic of viewer appropriation of pervasive public displays, providing an initial exploration of the space and then describing an architecture to support such content. In this chapter we have provided initial background to underpin and motivate the work.

We began by describing a recent research direction for public displays, in which it is suggested that to overcome 'Display Blindness' and encourage innovation, display deployments should become open to content from a wide range of sources. One potential content source is that of the display viewer themselves, we describe such content as *personalised* or *viewer-driven* content, and the process of placing that content on the screen as user/viewer *appropriation*.

We also described our existing testbed of networked displays in public spaces on the Lancaster University campus. This testbed is a permanent deployment in daily use and provides an excellent grounding for much of the experimental work described in Chapter 3.

In the remainder of this thesis we further explore the motivation for opening up public display systems to viewer appropriation. We consider a design space for user appropriation of public displays and present a number of scenarios and experimental works that explore different aspects of this space. We present an architecture for supporting appropriation within pervasive display installations and describe its implementation and evaluation.

Chapter 2

Background

2.1 Overview

A wealth of research and commercial effort has been invested in digital public display systems. This chapter begins (Section 2.2) with an overview of key research in the evolution of public displays, and a summary of the current state of the digital signage industry including key systems for commercial signage. We then survey how viewers respond to the display systems they encounter (Section 2.4). We highlight recent research that suggests typical passers-by do not attend to the digital displays in their environment due to low expectations of value for the display content.

In Section 2.5 we present a survey of research with contributions in the area of user appropriation/personalisation of displays, including traditional computing protocols for connecting to remote applications and displays. This survey is motivated by the audience responses summarised in Section 2.4 and by the argument that introducing new forms of content could increase the value of displays to their viewers (as presented in Section 1.2). The personalisation work presented in this thesis represents one potential future content source that may help to address the observed phenomenon of 'Display Blindness'.

The work in this chapter has also been published in:

- Sarah Clinch. Smartphones and pervasive public displays. *Pervasive Computing, IEEE*, 12(1):92–95, 2013 [Cli13].
- Nigel Davies, Sarah Clinch, and Florian Alt. *Pervasive Displays: Understanding the Future of Digital Signage.* Synthesis Lectures on Computer Science. Morgan & Claypool Publishers, In Press [DCAss].

2.2 A Brief History of Digital Signage

In this section we provide a chronological survey of public display research identifying some of the major themes prevalent at each time to help structure the survey. We do not intend to imply that these were the only topics of interest during the periods to which they are connected, nor that all work on a topic occurred within the connected time-slot. However, we do hope that the combination of times and themes helps the reader to understand the pervasive display research landscape.

2.2.1 1980s & Early 1990s – Media Links

The use of digital displays in public spaces first emerged as a topic for research in the 1980s. Early studies took the form of 'media links', using video and audio links to connect together physically separate spaces.

Kit Galloway and Sherrie Rabinowitz created "Hole-In-Space" [GR80], a three day art installation in November 1980. The installation featured two large back-projected displays (plus speakers, microphones and cameras) installed in sidewalk-facing windows of the Lincoln Center for the Performing Arts in New York City and "The Broadway" department store in Los Angeles. A satellite link between the two cities allowed the creation of a virtual window in which the video feed of New York was shown on the screen in L.A. and the video from L.A. in New York.

A set of follow-on projects explored longer-lived connections, using media links to facilitate interactions between workers in multi-site research institutions.

The Xerox PARC Media Spaces [BHI93, GA86] connected researchers at sites in Palo Alto and Portland by providing steerable video and audio links in the "common area" of each site. The media links ran 24 hours a day, seven days a week for over two years. Whilst originally intended to support formal meetings, the majority of interactions over the links were chance encounters lasting for less than five minutes. A similar system at Bellcore Labs, the VideoWindow [FKC90], connected researchers on two different floors of the building using large projected displays in common areas.

A second wave of media links on large public displays was seen during the early 2000s. The Microsoft Virtual Kitchen [JVG$^+$01] linked three workplace common areas (in this case, kitchens) using media connections. A projected image in each kitchen showed feeds from two other kitchens alongside the view from the local kitchen and a television channel designed to attract viewer attention. In response to privacy concerns, the system provided a mechanism to allow users to temporarily disable the camera and microphone. Telemurals [KD03, KD04] provided a more abstract link between two spaces in which video feeds from both connected sites were transformed to provide a single projected view. The Telemurals projection altered to reveal more detail when it detected interactions between the sites.

Media links remain an area of interest for modern artists; in 2008 Paul St George created the 'telectroscope', an outdoor interactive video link between London and New York [Art12].

2.2.2 Early 1990s – Ubiquitous Computing

In 1991, Mark Weiser published his seminal paper describing a vision for the future of "Ubiquitous Computing" in which display devices were particularly prominent [Wei91]. Weiser described three categories of device, 'tabs', 'pads' and 'boards', and illustrated how such ubiquitous devices could be used to support everyday working practices. Whilst the smallest devices ('tabs' and 'pads') are more representative of our current mobile devices and tablets, Weiser's boards took the form of large, wall-mounted displays that served a number of purposes, acting as collaborative working spaces, community bulletin boards, virtual book-cases and video screens. PARC's prototype board implementation provided a 1024x768 pixel screen space and a 40 x 60" display space that could be written on using wireless "electronic chalk".

2.2.3 Mid 1990s – Wearable Displays

While many early public display systems were static installations, perhaps the earliest work on movable displays came in the form of small interfaces that could be worn on the body and used to display information to others. One early system, ThinkingTags [BMMR96, BMRS98, Bor02] (part of the GroupWear project) extended conventional name tags by augmenting them with 5 coloured LEDs which allowed a viewer to see the degree to which they shared opinions with the wearer. Infrared communication allowed two badges to exchange answers to five questions – for each answer the two badge owners had in common a green LED would be lit, for each question they answered differently a red LED was lit. These small displays were intended to spark conversations between attendees at events.

Further work in the GroupWear project led to the development of Meme Tags [BMV+98, Bor02]. Like ThinkingTags, Meme Tags took the form of a wearable name tag. Each tag included an LCD screen capable of showing up to thirty-two characters of text at any time. Infrared communication was again used to support data exchange between the tags – when badges came into range each would display a meme for the other using the name of the viewer e.g.

```
Fresh meme for Bob:
Computing should be about
insight, not numbers
```
(Borovoy et al. 1998 [BMV+98])

Like many modern systems (see Section 2.2.7.2), Meme Tags also provided support for information takeaway – upon encountering a meme on another's tag, viewers could transfer that meme to their own to allow further sharing.

Ljungstrand et al. created their wearable display platform, 'WearBoy' [LBF99], by adapting components from a Nintendo GameBoy. Rearranging the components of the gaming device allowed them to create a small wearable device just larger than the 2.6" colour screen and weighing less than ninety grams. Whilst much wearable computing work has focussed on providing computing for the wearer (e.g. by augmenting reality with an additional information feed) [FV10, HLSH09, WFC06], the WearBoy platform formed the basis of two wearable display devices designed to be visible to those other than the wearer. Active-Jewel [LBF99] provided a mechanism for users to express themselves with digital jewellery - the WearBoy device formed a digital brooch that supported moving, repeating and non-repeating patterns as a mechanism for drawing attention and decorating the wearer with a visual representation of their personal values. By contrast, the BubbleBadge [FB99, LBF99] was intended to be open to content from the viewer and the environment as well as the badge wearer; scenarios focussed on the display of information in the form of relatively short chunks of text. The badge was intended to be worn as a brooch, close to the face, with the intention of providing information to the viewer and promoting face-to-face interactions.

Research in the wearable computing space continues but is predominantly focussed on providing personal computing services to the wearer. For future wearable displays, advances in smart textiles have been demonstrated to provide display capabilities [Bal02, CMKK11, Gou03, R&03].

2.2.4 Late 1990s – Ambient Displays

Inspired by Mark Weiser's vision for ubiquitous, invisible, computing [Wei91], a series of projects in the mid to late 1990s explored methods of ambiently augmenting the environment with information displays.

2.2.4.1 Novel Ambient Display Hardware

Mark Weiser's vision for ubiquitous or 'calm' computing [WB97], was explored in a number of projects that focussed on methods of peripherally displaying information through non-traditional hardware.

Natalie Jeremijenko's Dangling String [WB97] (also known as 'Live Wire') comprised of an 8 foot piece of plastic spaghetti (string) that hung from a stepper motor connected to a nearby ethernet cable. As data was transmitted over the network, the electrical signals caused the motor to turn resulting in movement of the string and yielding a peripheral indication of the level of traffic. During high-traffic periods the string would whirl round “madly”, accompanied by an audible noise from the motor, whilst in quiet periods only a small twitching movement would be visible.

Bohlen and Mateas’s Office Plant #1 [BM98] took the form of a robotic structure intended to be reminiscent of a small plant. The plant was comprised of a small spherical bulb mounted on a stem and surrounded by wire fronds; a speaker was concealed within the bulb. The plant responded to incoming email by altering its physical posture and could also emit a variety of background noises. The idea of ambient plant displays was explored further in a set of projects in the mid 2000s: the LaughingLily [AS03] provided an artificial plant mechanised to reflect the types of conversation occurring in a meeting room (silence, productive conversation, argument), whilst the Living Plant Display [HKH+04], Spore 1.1 [Eas04] and PlantDisplay [KW06] manipulated the environmental conditions of real plants such that their health and growth reflected some additional data sources (e.g. watering the plant to reflect an increase in share prices [Eas04] or denying the plant light to reflect a lack of social interaction between the plant owner and another individual [KW06] or directing light, and therefore plant growth, to reflect trash/recycling disposal [HKH+04]).

MIT’s tangible media group developed Pinwheels [DWI98, WID+98, IRF01] in which folded fibreglass pinwheels were arranged in arrays, each powered by a motor. Also designed by the same group, the Water Lamp [DWI98, WID+98] used a light bulb and solenoids mounted on a water tray to create projected ripple patterns of light and shadows on the ceiling. Collectively the two devices were described as ‘Ambient Fixtures’ and were a continuation of an earlier work on the ‘ambientROOM’ [IWB+98, WID+98]. Both Pinwheels and the Water Lamp were used within the MIT lab to display network data: wireless LAN traffic was shown on the Pinwheels and web hits on the Water Lamp. A Pinwheels deployment in Tokyo was also used to display data about a range of activities including elevator usage, email traffic, and motion at streets and intersections.

The Information Percolator [HHT99] used a series of 32 transparent tubes filled with water in order to create a scrolling display of approximately 32 x 25 bubble pixels. Each tube was fitted with two aquarium pumps that enabled air to be released up the tubes in a precise manner in order to create a bubble to travel through the water. As the bubbles rose up the display it created a natural scrolling effect. The display could be used to display short pieces of text, a simple representation of activity in front of the display or in a nearby corridor, and to gain attention through audio and visual patterns in order to notify those in the space of an upcoming event or of the passage of time.

Rodenstein experimented with use of Privacy Film-covered windows as a surface for a projected images and animation to allow peripheral display of short-term weather forecast data [Rod99]. Their use of windows in this way was intended to complement people's natural instinct to look out for information gathering and aesthetic purposes. Like Rodenstein's windows, later work on the FogScreen [RP02, RP04, RDO+05, RP05, REE+06, RL07] used projection to provide a mechanism for traditional image and animation display on an unconventional medium. A variety of tracking mechanisms were trialled to enable interactivity with the FogScreen [RP04, RDO+05, RP05] and a range of applications suggested including performing arts [REE+06] and advertising [RL07].

Hello.Wall [PRS+03] (also referred to as GossipWall [SRP+03]) was based around a non-standard panel display comprised of 124 clusters of LEDs (cells) combined with short-range transponders that allowed communication with a ViewPort mobile device. The wall displayed information through a series of light patterns which could only be decoded using the mobile device. Detection of the ViewPort devices also allowed the system to identify the number and distance of nearby viewers, allowing different information display for users in an 'ambient zone' (not detectable at the display), 'notification zone' (detectable only with long-range readers) and 'interaction zone' (close enough to interact with a cell on the wall). Within the interaction zone, viewers could also use a ViewPort to draw on the display by lighting-up cells on the wall.

More recently Breakaway [JFHZ05] and Clouds [HDM+10, RHM+10] designed ambient information displays as a mechanism for changing viewer behaviour. Breakaway [JFHZ05] was a small desk sculpture that moved into a slouching pose to reflect the inactivity of a desk occupant; once the worker took some time away from their desk, the sculpture would return to an upright position. The Clouds installation at the Open University [HDM+10, RHM+10] was part of a larger project designed to encourage use of the stairs in preference to the elevator when travelling within a particular building. Twenty-four coloured spheres hung from the ceiling: half were orange and represented elevator use, the remaining half were grey and represented use of the stairs. Each set of twelve spheres could be moved closer to the ceiling or floor to reflect the changing use of stairs and elevator; vertical distance between the Clouds would indicate the difference between the number of people taking the stairs versus those taking the elevator. The installation was supplemented by an array of plasma screens that gave a more detailed representation of stair/elevator usage for the current or previous working week.

2.2.4.2 Artistic Information Visualisations on Traditional Displays

Ambient displays have also taken the form of 'informative art' shown on more traditional digital display media (typically large LCDs). Like those on novel hardware, such systems typically aimed to provide non-intrusive, aesthetically-pleasing data representation.

Redstrom et al. [RSH00] developed a number of prototypes to explore the concept of informative art, a method of presenting changing information as artistic composition. Web Aware [RSH00, SH00] used a mapping technique to visualise visits to pages within a corporate website. Each document available on the site was represented by a dot shown on a public display within the office environment (each dot was positioned relatively such that nearby dots reflected relationships within the site structure). Each document request resulted in highlighting of the corresponding dot which gradually faded away over time. The resulting visualisation provided a peripheral information source that resembled a picture of a galaxy. A six-month deployment of the system on a variety of display technologies (projectors, small and large wall-mounted flat panels) showed that the system was effective as a peripheral information source and provided a "quite beautiful" conversation piece [SH00, p. 2]. The Klein Clock and De Stijlistic Dynamics both visualised email traffic in a manner that was intended to closely reflect the work of a well-known 20th Century artist [RSH00]. For example, De Stijlistic Dynamics showed the relative frequency with which a set of individuals sent and received email on a geometric composition inspired by the work of Piet Mondrian. Each user was represented by a coloured rectangle which grew in size as emails were sent/received (during periods where no messages were exchanged, that user's block would decrease). Other prototypes included a clock that displayed the current time as a block of colour (lightest at noon and darkest at midnight) and a projected display, ChatterBox, that generated sentences based on fragments from recent email messages [RSH00].

The IBM Fishtank [Far01] was an electronic fishtank representation displayed on a 50" touch-screen in a shared lab space, in which entities within the office (persons, objects) were represented by customisable fish. Observing the appearance, movement and speech of a fish in the Fishtank could reveal information about a person's interactions with others in the office or the busyness of their surroundings. Alternatively the fish could be used to describe the state of some other entity, such as a stock value. Users could customise their fish's appearance, speech responses (shown as speech bubbles and used to respond to simple queries), and social behaviour (for example, choosing to swim towards touch-events or to move away). The authors reported immediate interest in the system with many of the lab occupants choosing to create fish, however no formal evaluation was reported.

Holmquist and Skog describe four informative art prototypes: Weather Composition, Stone Garden, Soup Clock and Motion Painting [HS03]. Like De Stijlistic Dynamics, the Weather Composition was inspired by the paintings of Piet Mondrian. Six coloured rectangles were displayed over a grid of black lines, each representing the weather in a different international city. The position of the blocks was intended to roughly correspond with geographic location, with the Greenwich meridian placed in the centre. The colour of the block indicated whether the weather in the city was clear (yellow), cloudy (red) or wet (blue), whilst the size indicated the temperature (larger shapes for hotter conditions). A similar geographic layout was used in the Stone Garden, which represented recent seismic activity by placing a stone in the appropriate location; different stone images were used to represent points on the Richter scale. The Soup Clock provided a countdown timer in the form of a tiled display of soup cans, similar to images produced by Andy Warhol in the early 1960s; as the time elapsed, the image would move from being entirely yellow (asparagus) cans to being increasingly dominated by red (tomato) cans. Finally, the Motion Painting represented the level of activity (based on a camera image) in a space over time. The application continuously drew thin vertical lines in changing colours from left to right. Lines that were similar in hue represented similar levels of activity whilst greater differences in hue indicated greater changes in activity levels. The four prototypes were trialled at SIGGRAPH 2001 Emerging Technologies and were projected onto pieces of white fabric. The authors reported that, whilst some initial explanation was needed to interpret the images, visitors quickly understood the installation and were able to explain it to others [HS03].

Providing a visualisation of activity was also the aim of Skog's Activity Wallpaper [Sko04]. Their projected prototype was designed for a local café and displayed activity levels in the café over the last week. A set of dots were displayed in an 8x30 grid of cells, one column for each day and thirty rows each representing a short time period within the day. The number of dots in each cell indicated the busyness of the café at that time, whilst the colour of the dots represented background noise levels. Within the e-Campus deployment (see Section 1.4), an outdoor artistic installation called Metamorphosis represented activity levels in a more abstract manner [SFD+06b, The05]. Three projectors created a large display upon which a video of butterfly was displayed. Events triggered by passing traffic and other environmental changes altered the behaviour of the butterfly, for example causing it to appear to take fright and fly away from traffic.

Vogel and Balakrishnan [VB04] identified eight design principles for an interactive public ambient display: calm aesthetics, meaningful content, socially acceptable communication, short duration interaction, immediate usability (learning by exploration, demonstration), multi-user, combination of public and appropriate personal information, and privacy. The authors also highlighted the importance of a smooth transition into user-interaction phases, and designed a prototype office display designed to comply with the identified design principles. Their system comprised of a 50" plasma screen with touch sensitivity and motion tracking. Ambient (default) content was represented by four horizontal bars across the display that provided an abstract representation of the weather, activity levels in remote offices, an appointments calendar and messaging patterns. As users interacted with the display, the ambient content was augmented with an intersecting vertical bar onto which personal information could be overlaid. An informal user evaluation focussed largely on the interaction phases and found that most of the users quickly understood the interaction methods. During the evaluation, some users also commented positively on the ambient nature of the content, and the way content transitioned from ambient to interactive content in a "pleasing, non-direct way" [VB04, p. 9].

More recently, digital projectors have become an increasingly common tool of urban artists (for example, URBANSCREEN have projected images onto the Sydney Opera House; Rice University, Texas; and the Leopold Museum, Vienna [URB13]). Researchers have also proposed the use of personal mobile projection devices to allow users to create their own transient, public displays in the form of dynamic, collaborative art for self-expression [SJE11].

2.2.4.3 FLUMP

The Flexible Ubiquitous Monitor Project (FLUMP) [FWDF96] used traditional form displays with the aim of providing a "heterogenous, ubiquitous multimedia information system... enabling useful information to follow people around the [departmental] building" [FWDF96, p. 1].

The initial FLUMP deployment consisted of a single "FLUMP station", a wall-mounted CRT display with an associated hidden computer [FWDF96]. A second station was later added [FDE12]. In line with Weiser's vision of seamless, transparent computing, the displays were placed at head height to minimise disruption and avoid unwanted attention.

To allow content to follow users as they passed the displays, FLUMP used an infra-red badge and scanner system – Olivetti's Active Badges [WHaG92]. If a FLUMP station detected a user's Active Badge it would show that user's personalised homepage. Homepages could include a variety of content items including unread email messages for the user, upcoming appointments, a cartoon, and the opening status of a local coffee bar. When no registered users were detected to be in the vicinity of display, the FLUMP station would carousel through a sequence of default web pages.

2.2.5 Early 2000s – Displays in Office Environments

2.2.5.1 Door Displays

Throughout the early 2000s, researchers began to explore the use of digital displays as doorplates for offices, meeting rooms and shared spaces. Perhaps the first such system, Palplates [MS97], was deployed at FX Palo Alto Lab and consisted of a set of touch-screen terminals whose interfaces were keyed to their specific location within the lab and were intended to support common tasks. Prototypes by Nguyen et al. [NTDM00] used small displays to support location specific messages and information (e.g. the last known location of a particular office inhabitant); their 'Dynamic Door Displays' would also allow a viewer to leave behind a message for the sign owner.

Outcast [MCL01] consisted of a single flat-panel door display intended to show content both peripherally (cycling through content selected by the owner) and in response to active interaction by passers-by. Popular content items included the owner's location information and calendar but unlike other systems of this type the system for messaging the owner through the display was not popular – viewers of the screen did not consider the feature to be reliable and so continued to leave post-it notes rather than digital messages.

Two generations of Hermes door displays were developed and deployed at Lancaster University. Hermes I [CFD03, CFDR02, FCFD04] was deployed outside ten offices for twenty-seven months between 2001 and 2004. PDA devices were mounted outside offices within the Computing department. Office owners could use their displays to leave a message for passers-by (with additional specific messages for particular individuals as identified through Java iButton authentication): messages could be entered on the devices themselves, through a web interface, via SMS or MMS, using tangible buttons or via email. Visitors could leave messages at the device using a stylus and touch-screen – their messages could then be retrieved by the owner locally at the device, or remotely using a web interface or email client. Following relocation of the department to a new building, the Hermes II [CTF+12] system featured forty displays with wider screens than the original deployment. In addition to the original Hermes I features, the devices supported video and audio messages and were designed to support multiple-owner displays for shared offices. Additional functionality was added for indoor navigation using the GAUDI system [KKK05, KCF+06] – a portal display could be used to allow users to enter a destination, each Hermes display en-route then each showed an arrow directing the user to their destination.

The Intelligent Mobile Messaging System (IMMS) [BJ04], was deployed outside staff offices within the Computer Science department at the University of Birmingham. The system used handheld devices fitted to office doors to act as information and messaging terminals for students who were hoping to interact with the office occupants. Office owners could update the message on their device using SMS or a web interface and messages left by students could be received through the same mechanisms (depending on the urgency of the message and the device owners' preference).

Unlike the previous systems in this section, RoomWizard [OPL03] was designed for deployment not on personal offices, but shared meeting rooms. Each RoomWizard display consisted of an eight inch colour touch-screen mounted outside a bookable meeting room; each display also had a pair of light strips along the side of the casing. The interface would display the timetable of bookings for the day, a number of lines of text describing the room's current status (e.g. "booked for Person X") and also a coloured light indication of the room's availability (green lights indicated availability and red that the room was currently unavailable). Advance bookings could be made through a web interface (each display also acted as a web server) but ad-hoc bookings could be made in-situ if the room was available. A deployment of five RoomWizard displays was trialled in two buildings of a large UK company. Such systems are, of course, now commonplace.

2.2.5.2 Workplace Awareness

In addition to the use of small public displays as digital door plates, the early 2000s also featured a significant body of work exploring the use of digital displays for improving awareness and developing a sense of community within the workplace. An early work, the Learning Communities Newspaper [HBL98] took the form of a web-based application projected in a shared space used by members of the Learning Communities group at Apple. News stories were submitted by group members, via email, to inform other members and guests about their project work and events.

Another shared workplace display, the Notification Collage [GR01] allowed its contributions to be submitted through a set of desktop client applications (e.g. a video generator or sticky note creation tool). A single large smart board in a laboratory common area displayed the collage of contributed items.

Greenberg described a prototype system, Dynamic Photos [Gre99], in which a group photograph shown on a touch-sensitive display could be used to promote awareness of the availability of the photograph's subjects and to support transition from awareness to conversation by providing a live video connection. Greenberg used the elastic presentation system [CCF95] to distort a digital image of the group such that the size of a person's face indicated their availability (as detected through sensor data). To engage a member of the group in a video conversation, a person could walk up to the photograph and run their finger over the portion containing the person that they wished to talk to.

The Aware Community Portal [SWS01] at MIT Media Lab consisted of a projected display with an associated camera and server used to display items of relevance to researchers within the laboratory. The display showed live news and weather feeds, an hourly cartoon strip, a periodic clock update, and a feed from the camera. Unlike many similar displays within research workplaces, the Aware Community Portal altered its behaviour in response to viewer engagement; as a user walked past the display, articles would cycle through in sequence, if a user stopped to look at the display the cycle would pause and more detailed coverage of the current article would be shown. In order to promote awareness of other people's behaviour within the space, the display would show captured images of those looking at the articles on a timeline alongside the articles themselves and would also show a general overview of movement within the space.

The CommunityWall [GMRS03, SG02] displays deployed at the Xerox Research Centre used touch-sensitive SMART Boards [SMA13] to display items submitted by users. Items could be contributed through a variety of methods including email and a web bookmarking system. The display would show 10-15 items at a time selected based on a set of rules that assigned each item a priority.

The touch interface on the display also allowed viewers to expand, email, print, rate and comment on each item. Evaluation of the deployment of two displays showed that the system had approximately twenty regular users and that at least 50% of articles were interesting enough to users to have received some viewer interaction [GMRS03].

The MessyBoard was originally designed with the goal of improving memory and awareness [FFP02] but later development focussed on improving communication between coworkers [Fas04]. The MessyBoard was a shared bulletin board space that could be modified via the web and viewed as a screensaver or projected display. Trials with a number of different groups showed that the MessyBoard was used both for work-related purposes (e.g. arranging meetings or collaborating on projects) and for play (e.g. collaborative games).

The Plasma Poster Network [CNDG03] consisted of three touch-enabled plasma displays. The displays were located in the FX Palo Alto Laboratory (one in the kitchen, one in a foyer and one in a hallway) and were oriented in a portrait format to reflect the typical layout of traditional paper posters. Content was generated from the laboratory's intranet pages and from items submitted by users via email and the web. A "PosterShow" interface cycled through content items and touch-screen buttons were provided for navigating between content items, printing, forwarding and responding to items. The Plasma Poster Network was evaluated through examination of the data collected through ten months of use and through interviews and email surveys. Over the ten months 859 items were submitted, users commented that they tended to submit items they thought would be of peripheral interest to others. The authors commented that their observations indicated that the displays had increased social interaction between members of the lab.

Huang et al.'s Semi-Public Display [HM03] was deployed in the Everyday Computing Lab at the Georgia Institute of Technology with the intention of promoting collaboration and coordination of a small co-located group. The system used a touch-enabled SMART Board [SMA13] and provided four applications: a 'collaboration space' for asynchronous brainstorming and discussion around a topic; a set of reminders; an 'active portrait' that provides a graphical representation of group activity over time; and an 'attendance panel' that provides an abstract visualisation of planned attendance at upcoming events. After two weeks of deployment, an initial evaluation suggested that the attendance panel and reminders were regarded positively and were useful in helping maintain awareness within the group whilst the collaboration board and active portrait were less useful, perhaps because of technical problems.

IM Here [HRS04] was designed to support a shared instant messaging (IM) service on digital displays within a work environment, based on the observations that IM could be used to help with work tasks and that many individuals spend significant portions of time away from their personal workspace (e.g. for formal and informal meetings, collaborative activities). The messaging client supported broadcast messages as well as those directed to individuals, and all messages were sent from a special 'IM Here' message account which ensured the recipient was aware of the public nature of the communication. In addition to supporting instant messaging, the system also provided a cycle of event flyers highlighting both formal and casual office events. Over the first six weeks of deployment, at least eight unique individuals used the system in forty-one sessions, sending 165 messages to seventeen unique recipients (not including broadcast messages). Interviews with office inhabitants indicated that the system was generally considered to be a positive addition to the office; those that had not used the system typically attributed this to a lack of cause to do so and stated that they would like to use the system in the future.

The Hermes Photo display [CDF+05a, CDF+05b, CTF+12] was first deployed at Lancaster University in June 2003 and consisted of a single wireless touch-screen display in a corridor of the Computing department. The screen cycled through photographs contributed by users through MMS and email. Feedback received regarding the deployment indicated that those on the corridor felt their sense of community had increased as a result of display use [CTF+12]. Following transfer to a new building, the display was located in a more public corridor area and new features were added, users could now contribute photographs using Bluetooth and download pictures from the display using the same mechanism. The process of pairing and then sending or receiving a 250 Kbyte file would take approximately one minute [CDF+05a]. The system was evaluated with a user study of seventeen participants; the majority were satisfied with the usability of the Bluetooth interaction between their mobile phone and the display and were positive about the impact such a display could have on their sense of belonging to a campus community.

Congleton et al.'s proactive display prototypes, Prospero [Con07], C4 [MCH08] and ProD [CAN08] were also deployed within academic environments. Each system was designed to support content based on preferences from nearby users. A set of user-customisable display inputs were presented either one user at a time (Prospero) or arranged into collages combining multiple users' content (C4 and ProD). An evaluation survey issued after the first four weeks of the C4 deployment indicated that users generally considered the displays to have had a positive impact on their relationships with others in their workplace. Evaluations of Prospero and ProD were not reported. Prospero, C4 and ProD are described in more detail in Section 2.5.7.

Moving away from the academic setting, Bardram et al.'s AwareMedia system [BHS06] was designed to raise awareness and support messaging within the surgical ward of a hospital. The deployment consisted of a total of ten display clients (mostly large touch-screen displays, some with associated cameras) and Bluetooth-based location tracking. The screens were required to be very information-heavy, with support for social awareness (awareness of a person), spatial awareness (awareness of a place) and temporal awareness (awareness of past, present and future) as well as providing a messaging service between locations. Evaluation interviews three months into the deployment suggested that the displays were useful for asynchronous communication and that the awareness information was useful for informing behaviour. Wilson et al. [WGF06] also explored use of a large display in a medical setting. They focussed on handover practices between shifts and ran a simple two-week probe in which a digital photograph of existing paper records was projected onto the wall of the handover room.

2.2.6 Mid 2000s – Promoting Social Interaction

Following on from themes introduced by early wearable display research [BMV$^+$98, BMMR96, BMRS98, Bor02], a number of display projects in the early to mid

2000s had the specific aim of promoting social interaction.

The GroupCast system [MCL01, McC02, McC03] consisted of a single peripheral display deployed in a common area of the Accenture Technology Labs. Unlike other workplace displays, its aim was not to promote awareness within the department nor was it intended for active engagement. Instead the display aimed to show items of interest to at least one passer by (as identified by an infrared badge system) in the hope of sparking conversations between colleagues.

The Interactive Wall Map [McC02] consisted of a large wall map (approximately 4m x 2.5m) that was augmented with three pairs of flat-panel touch-screen monitors placed within different geographic regions, and twenty-four switches placed over cities of potential interest. Location related information was displayed in response to user presence or could be explicitly requested by using the switches. The map was intended to elicit conversation and stories around place and travel.

Brignall and Rogers' shared display, The Opinionizer [BR03], was designed for informal gatherings and allowed users to add their views and opinions for others to observe and comment on. The system was trialled in two scenarios: a book launch party (two hours, approximately 300 attendees) and a welcome party for new postgraduate students within one school of a university (2.5 hour deployment, approximately 150 attendees). The Opinionizer consisted of a large projection controlled by a laptop which was also used for entry of the opinions/ comments. Interaction with The Opinionizer increased throughout the trials and many participants were positive about the display as a mechanism for supporting social interaction.

The BlueBoard [RG01, RDS02, RTW02, RTD04] was a large plasma display designed to support both rapid, individual interactions and small group collaborations. Personal content was accessed through use of a p-con, an image representation of a user currently 'badged-in' to the device. When not in use, the BlueBoard looped through a set of location-determined pages. Social use of BlueBoard was investigated through use of a field study with 163 participants leading the authors to note six prominent social interaction patterns: learning through observing others, turn-taking, emergence of a 'driver' who interacts on behalf of the group, the use of in-situ side-channels for information channels, an unwillingness to reach into another person's interaction space and an uncertainty caused by a lack of established etiquette for multi-person interactions.

The Dynamo interactive surface [BIF+04, IBR+03] consisted of one or more displays, tiled either horizontally (a standard multi-display configuration) or vertically (creating a tabletop-style surface and accompanying display). The surfaces were designed to promote collaboration and could be used as a communal resource with private areas 'carved off' for individual or shared use. Interaction with Dynamo could be achieved through a mobile device (e.g. laptop, PDA) or through an 'interaction point' consisting of a set of USB slots, a wireless keyboard and wireless mouse. The surface supported a wide-range of media types that could be displayed or exchanged. A two-week deployment in a high school common room showed that the system was an effective means of sharing media both by making the files available and also by promoting performance-style interactions [BIF+04]. The students quickly appropriated the system for their own purposes.

Two projects supported social interaction opportunities in a café environment. Schminky [RLHS04] was a music-based game that ran over a set of handheld devices and a large public display. Each player used a handheld device and headphones to listen to a set of tracks and identify which sound was 'missing' at a given point in time. Individual and networked (multiplayer) games were both supported – a multi-player game could be started by any player and resulted in a broadcast request for players being sent to every device. A public display showed the social network created by players (who had played whom). 74% of players took part in a group game and 48% of those playing group games played with strangers. Increased movement between tables was also observed as players used it as a "portable [social] networking device" [RLHS04, p. 3]. Jukola [OLJ+04] allowed users of the café to nominate tracks to be played (similar to a traditional jukebox) through a touch-screen display. Four nominations would be shown on a handheld display on each café table, and users could vote on the track to be played next. Unlike similar systems like MobiLenin (see Section 2.2.7.2) which used individual's personal mobile devices for voting, the shared nature of both the large display and the handheld devices encouraged social interaction. Comments from a week-long field trial indicated that voting was conducted as a group and that the system often triggered conversations about the nominated tracks.

Whilst both systems typically increased social interaction, some negative effects were also noted. Schminky's headphones and cognitive demands could isolate individuals from their surroundings whilst the conversations generated by Jukola did occasionally distract from other discussions.

Ticket2Talk [MMS+04] was deployed at an academic conference to provide opportunities for starting conversation. Each participating user was equipped with an RFID tag and registered by submitting an image and caption representing an interest or topic that they would be happy to talk with others about. A single Ticket2Talk display was deployed behind one of three tables used for serving drinks and snacks during breaks; as tags were detected at the display they were added to a queue of nearby participants. The display cycled through the queue displaying each person's conversation starters alongside their name and photograph; each one was displayed for five seconds. Evaluation suggested that the display did result in some additional social interactions but that overall respondents did not consider that Ticket2Talk made a significant positive or negative impact on the conference. Another conference system, Sparks[CLS+05] used projection to create an aura of interest keywords around individuals. Users entered their interests ahead of time and were encouraged to find others with similar interests through projected paths that join similar auras.

Displays continue to be used as a mechanism for promoting social interaction. Farnham et al. deployed their CoCollage display in a community-oriented café to support awareness and face-to-face interactions [FMP+09]. CoCollage provided tools for the creation of online profiles which allowed the sharing of media and provided a mechanism for online conversations. Café community members could then share items using their profile which were visualised as a continuously updated collage on the screen in the café – items from physically present users were prioritised over those from other community members. Within the first month of deployment, 82 users had created accounts. 71% of users uploaded content for sharing. Between 20% and 30% of users added comments to shared items, commented on profiles and sent instant messages. Users who reported that they would like to make friends were more likely to actively participate in the system.

Another recent deployment, a display-based yearbook system referred to as USIAlumni Faces [RML11], was found to support the sharing of memories and to stimulate conversation between groups of people meeting around the display. The system used a large screen together with a custom-built input device constructed from a Wii controller and infrared pen. Over two hundred attendees used the deployment of USIAlumni Faces at a university reunion event, successfully interacting with the system with no verbal instruction from the researchers.

FunSquare [LMEA11, MEL11, MLA11], used automatically generated fact snippets (referred to as autopoiesic content) shown on a UBI-hotspot display (see Section 2.2.7.1) to stimulate conversations similar to those observed when strangers meet in a space with an unusual feature (e.g. a fountain or sculpture). The FunSquare application featured an ambient mode in which the fact snippets were displayed and a game mode in which the facts were presented as a multiple-choice trivia question. The content was positively received and was observed to act as a conversation trigger [MLA11]. The FunSquare deployment contributed to the development of the 'interacting places' framework for displays that promote community interaction and place awareness [MLR+12, MLA12a, MLA12b, MLR+12]. The framework covered four key elements: stakeholders, communication channels, awareness diffusion and content viewers. The Interacting Places Framework was developed using an action research approach through four studies: a study of analog display use [AME+11], a study of ICT-based communication practices [MLR+12] as well as the deployment of FunSquare and Digifieds (discussed further in Section 2.2.7.2).

2.2.7 Current Research

2.2.7.1 Long-Lived Deployments

The late 2000s saw an emergence of long-lived deployments in a variety of locations including city centres, rural communities and university campuses.

Commercial display deployments, typically used for advertising purposes, became (and remain) increasingly common in urban spaces. For example, the INFOSCREEN deployments in Germany [Str13] and Austria [INF09] first began during the late 1990s, with expansion into Poland in 2009. Similar networks now exist worldwide (e.g. DSN [Dig12] has over a thousand displays across India, and Infoscreen Networks plc [INF12] owns and operates displays within public transport and urban areas of Kuala Lumpur, Malaysia.

The following paragraphs detail research deployments that may be considered ‘long lived’ – typically those deployed for a number of years. Within an urban context, many now feature as part of the street furniture. For deployments in rural areas and universities, the displays have often become a key tool for the community.

Urban Deployments

The first BBC Big Screen [BBC12] was deployed in Manchester in 2003. As of January 2013, a total of twenty-two large displays (plus associated sound systems) have been deployed in cities across the UK. The 25-square-metre displays are controlled by the British Broadcasting Company (BBC [BBC13a]) and are used to show a variety of local and national content including interactive games and coverage of significant events.

CityWall [Hel13, JMR+10, PSJ+07, PKS+08, MJP08] was first deployed outdoors in Helsinki in May 2007. A replacement interface was launched on a 2.6 metre wide display in October 2008 [Gal08]. The display was originally intended to show information during large events (e.g. the Eurovision Song Contest) but formed the basis for a number of multi-user interaction studies.

The UBI-hotspots deployment in Oulu, Finland [HLO+10, OKL+10] is one of the largest city centre research display deployments and consists of twelve hotspot sites each including one or two 57" LCD panels (indoor hotspots feature a single outward facing display whilst outdoor hotspots use two back-to-back LCDs to support use from both sides). Each hotspot is also equipped with a loudspeaker, cameras, network access points and an NFC reader. The hotspots were deployed in 2009 and have formed the basis for a number of significant studies, acting as a “‘heavyweight’ urban probe” [HLO+10, p. 1], and providing a platform for the annual International UBI Challenge event. Research supported by this platform covers a wide variety of topics including content generation (e.g. FunSquare described further in Section 2.2.6), interaction methods (e.g. BlueInfo and Digifieds both described in Section 2.2.7.2), interaction behaviour (e.g. Ojala et al. [OKK+12]), stakeholder values (see Section 2.2.7.3), and general guidelines for deployments e.g. ([HLO+10]).

In contrast to the previously described systems, the Campus Coffee Display [CTR+08, KGR08], was an indoor deployment and consisted of a single display sited within a café in Newcastle, UK. The display remained in place for over two years and provided information on local cultural events. For researchers, the display acted as a technology probe to explore display engagement behaviour.

Rural Deployments

In 2006, an adaptation of the Hermes photo display (described in Section 2.2.5.2) was deployed in Wray, a small community in Northern England [CTR+08, TCF+07]. The system was modified in response to community feedback and in 2010 the capacity to submit local advertisements, news and event information was added [CTF+12, TC10, TC12]. The display was positively received within the community eliciting a number of comments regarding its usefulness for new residents and visitors as well as existing residents [TC09]. Nnub [RB08, RB09], a similar display system, was deployed in a suburb of Brisbane in 2008 and has since expanded to include displays in a number of Brisbane communities [oT13a].

University Deployments

The initial set of e-Campus displays [FDE12, SFD06a, SFD+06b, Sto08] were deployed at Lancaster University in 2005 and formed a platform to support the university's digital signage needs as well as serving as a research testbed (see also Section 1.4).

A combination of 'News Displays' and 'Reminder Displays' formed the iDisplays [MPK07, MWE+09] deployment at the University of Münster, Germany. Following an initial prototype in 2005, a total of seven iDisplays were deployed in May 2006. Placement of either a News or Reminder display was optimised for the location type – News displays were deployed in entrance areas (highest number of unique viewers) whilst Reminder displays were located throughout the department building (for repeated viewing throughout the day). Four of the deployed displays were later reused for ReflectiveSigns [MEBK09]. Photos, comics, news, short videos and other content items were shown on the displays. Content display was initially displayed randomly, but as face recognition data was collected the displays combined historic view times with the known location of the display and the current time of day to weight the content by expected view time.

A three-part deployment was installed at the Open University in 2008. The deployment consisted of 'Clouds' (described in Section 2.2.4.1), 'Follow-The-Lights' (an LED navigation display) [HDM+10] and 'The History' (a tiled plasma display spanning 3m x 3.5m) [RHM+10]. Other long-term university deployments include the Hermes door displays (Section 2.2.5.1) and Instant Places (Section 2.2.7.2).

2.2.7.2 Smartphone Integration

Personal mobile devices such as PDAs, smartphones and tablets have been paired with public displays for a variety of purposes. We identify three common areas for research that combines the use of personal mobile devices with pervasive displays:

- Use of personal mobile devices as a method for interacting with pervasive displays, for control and data entry.
- Use of the mobile device as a mechanism for supporting the user in taking information away from the display.
- Combining mobile devices and displays to overcome the weaknesses of each: by foraging for a display a user can access more powerful and stable network and processing resources and increase screen real estate, whilst offloading some visualisation from the display to a mobile device can overcome privacy concerns and allow a personal view of the display content.

Using Personal Mobile Devices for Interaction

SMS/MMS/Voice Communication Whilst the term 'personal mobile devices' encompasses a range of appliances, the most ubiquitous is the mobile phone, and this is reflected by the wealth of research focussed on these devices. At their most basic, mobile phones provide an interface for textual input through SMS and voice communications.

TxtBoard [OHU+05] explored person-to-place messaging using an 8" fixed public display, for the home, with an onboard mobile phone. The display showed SMS messages that had been received together with the name and image of the sender. Viewers could scroll through previously received messages but no mechanism was provided for responding to the sender. A two-month case study of one family's use of the TxtBoard was conducted: all messages were logged and participants were interviewed regarding their usage. The authors identified four message types sent by family members to the board: explicit calls to action, informative/awareness messages, social etiquette (implicit but deniable calls to action), and social touch (e.g. encouragements). The authors also noted that despite awareness of the system, friends of the family did not send messages to the TxtBoard.

SPAM [CDF+07, FCK+04, GCFR05, GCR05], supported both person-to-place and place-to-place messaging using SMS. The deployment consisted of SPAM units (a touch-screen display plus associated GSM terminal) at two supported-housing sites in northern England. The system was deployed in October 2002 following a series of probes and discussions with potential users (care workers); small improvements were made during the early stages of the deployment (e.g. to allow messages from certain phone numbers to be discarded). An analysis of 360 messages (a sample that included messages from both sites at various stages of the deployment) revealed that approximately 22% of messages related to awareness (e.g. establishing presence, making others aware of news).

Erbad et al. developed the MAGIC Broker [EBF+08], a middleware framework that supported SMS, voice and web interaction between mobile devices and public displays. Tang et al.'s MAGICBoard [TFB+08] used the MAGIC Broker to support SMS interaction on their deployment at the University of British Columbia, Canada. Their two projected displays were intended to provide a forum for comments and votes on trivial topics (e.g. "Where would you rather be?" [TFB+08, p. 2]). Whilst a kiosk was provided for interaction, the authors hoped that SMS support might encourage participation from users who might avoid visible interaction methods (i.e. being seen using the kiosk) due to social embarrassment. Observational studies and log analysis showed that SMS users typically contributed longer, more carefully contributed content than kiosk users but were prone to formatting errors; however, the private nature of the SMS interactions meant that, unlike kiosk interactions, they did not draw others to the display (the so-called 'honey-pot' effect, described further in Section 2.4). The MAGICBroker was also used to develop Polar Defence, a tower-defence style game for large displays [FTLB08]. Viewers could place a set of towers by specifying coordinates over SMS.

The Hermes I door displays (described in Section 2.2.5.1) included SMS and MMS messaging in their input mechanisms, allowing users to set the message on their display [CFD03, CFDR02, FCFD04]. Likewise the Hermes I Photo display (Section 2.2.5.2) supported photo contributions via MMS [CDF+05a, CDF+05b, CTF+12].

Short-Range Communication: Bluetooth and NFC The introduction of short-range communication technologies into modern smartphones has provided another useful mechanism for interaction and data exchange between mobile users and public displays.

One of the earliest examples of Bluetooth interaction with a public display was also within the context of the Hermes projects. The Hermes II Photo display (Section 2.2.5.2) allowed contribution and takeaway of photos using Bluetooth [CDF+05a, CDF+05b, CTF+12]. Users were positive about usability of the method.

José et al.'s 2008 work on Instant Places [JOIH08] used Bluetooth device names as a mechanism for tagging features of their identity and to allow users to post items from their Flickr photostream. Like some of the advertising research described in Section 2.2.7.3, the system built a longitudinal profile for users over time based on the presence history of their mobile device's Bluetooth MAC address. The authors deployed two visualisations of user interactions: one that provided real-time representation of the current device names, and one that allowed historical tags to remain on the screen alongside current ones. During evaluation, Bluetooth scan data collected during the visualisation trials (three weeks for each visualisation) was compared with with some data collected immediately before the trials. The authors identified that the system prompted users to make their Bluetooth devices discoverable (percentage of visitors with discoverable devices increased from 4.7 to 7.0 percent) and to change their device names (average number of names per device increased from 1 to 1.5).

A similar technique was utilised by Mahato et al. [MKHS08], who used Bluetooth device names to support personalisation requests for displays and other environmental features. Users could encode their personal interests through a web form, generating a string of characters that could then be used as a Bluetooth device name (e.g. “bm+A1R3E5T3”). An online user study with 135 participants (72% German, 28% Indian) indicated that the system was most positively viewed when considered as part of a scenario for modifying background music in a café or for personalising the content of displays in a public transport setting. The study also highlighted cultural differences: 76% of the German participants considered the system a threat to their privacy compared to 29% of the Indian respondents.

BlueTone [DT09] combined use of Bluetooth device names, Bluetooth pairing and dual-tone multi-frequency signalling (DTMF) to allow users to perform complex display interactions using their Bluetooth-enabled mobile device. To use the system, an individual first changed their Bluetooth device name to indicate that they wished to control a display. A Bluetooth scanner at the display then discovered the user’s device and attempted to pair with it, connect the display to the device as an audio gateway. The audio profile used to connect the display and mobile device is then used to transmit the DTMF tones created as a user pressed keys on their device, the DTMF tones could then be decoded at the display and the keys mapped to predefined actions. Sample applications included YouTube (key presses could be used to alter the volume or pause playback) and Pong (where key presses could be used to move the paddle). No evaluation or user study was reported.

Hardy et al. [HR08, HRWP09] combined Bluetooth and NFC to provide a range of interaction operations including drag-and-drop and complex selection patterns (area selection, multi-selection). The authors projected a display over a 10 x 10 matrix of NFC tags. A Nokia 6131 NFC phone read the tags and forwarded the result to the display server using Bluetooth. A comparison with other interaction mechanisms suggested that user performance using NFC selection was considerably better than the same interaction using the phone joystick. Comparing NFC with touch interaction showed that whilst touch interaction yielded faster selection, NFC received higher usability ratings. Similar systems were developed by Vetter et al. [VHP+07] and Ramírez-González et al. [RMKA08]. Vetter et al.’s demonstration at Mobile HCI 2007 projected a map application onto a matrix of NFC tags [VHP+07], whilst Ramírez-González et al. proposed combining NFC tags and Bluetooth communication to enable touch interaction for development of new learning and advertising experiences [RMKA08].

Broll et al. [BGS+11] also combined Bluetooth communication with a projected NFC display. Their display projected onto a grid of twenty-four tiles, each composed of forty NFC tags – a much larger array of tags than in earlier systems. A 'Whack-a-mole' game was projected onto the display grid. A user study of the game with 18 subjects in both single-user and multi-user game play found that tag reading failed in 31% of cases resulting in ambivalent user responses to questions about the impact of the system's accuracy, speed and error rate on game play, and about its performance in comparison to other mobile phone-based interaction methods. Despite the failures, the users did find the game fun and easy to use. The multi-player aspect of the game was also well-received.

A summary of NFC interaction techniques for digital displays was published by Broll et al. [BRHW11]. They introduced a new interaction method in the form of NFC gestures. A prototype pinboard application was developed and eight different pinboard tasks were used to evaluate the different NFC interaction techniques. A single tag touch was the most popular form of interacting, the popularity of other methods was varied and highly task-dependant. The author's NFC gesture operations received good usability ratings for a variety of tasks but effectiveness ratings were also task dependant (e.g. gestures were rated an effective method of opening a context menu but were considered ineffective for opening and closing pinboard items or switching between pinboard views).

NFC and Bluetooth communication have also been used to provide data exchange to support the cyber-foraging of displays from a mobile device (e.g. Pering et al.'s Elope [PBW05] and Nickelsen et al. [NMS10], both described in Section 2.2.7.2).

Finally, Dementyev et al. [DGT+13] uses the energy and data from an NFC-enabled phone to update a bistable display tag. They developed a prototype Android application to screenshot a phone's contents and send it to the display over NFC. Transfer of the 5.67 kilobyte image to the display tag is completed in 3.4 seconds during which enough power is harvested to be able to update the display with the received image.

Mobile Internet Connectivity Internet connectivity is an increasingly ubiquitous feature of modern mobile devices. Since the early 2000s, technologies such as WAP and WiFi have provided a useful mechanism for communication between a mobile user and the public displays they encounter. For example, the WebWall software framework allowed users to request content onto a public display by submitting a request from their mobile device through SMS, WAP and WiFi [FV02]. Applications developed for the WebWall included video and picture galleries, polls and an auction system. No evaluation was reported.

MobiLenin [SO05] used GRPS connectivity to allow users to interact with a music video voting application. The application, shown on a large public display, indicated the start and end of voting periods. Votes were cast through a custom mobile application that used GPRS to connect back to a voting server. The outcome of the vote and the resulting music video were then shown on the display. A trial in a real-world restaurant setting with fourteen participants found that users found the mobile voting mechanism to be "fun" and that they would "[look] forward to more interaction like this". Suggested applications from users included avatar creation and trivia games.

Flourish [FP12, HSS13] used an HTML5 web application to allow users to add a flower to a display installation showing a meadow full of flowers. The application was made available at the public display through use of a QR code and restricted flower submission to those within 10 metres of the display.

The development of mobile websites offers potential for display interaction within the native browsers of a mobile device. For example, the 'Public Display for Human Rights' allowed users passing a display to contribute their opinion on human rights cases shown on the display [FP12].

Light- and Camera-based Interactions Small portable cameras are popular both as standalone devices and as an integrated feature of modern mobile phones ('camera phones') and tablets. Such cameras provide a useful tool for interacting with displays. Related research has also used light-emitting features of such devices (e.g. a camera flash, other flashlight or LCD screen) as a display-interaction mechanism.

Rohs and Gfeller [RG04] developed a system for interacting with a variety of real-world objects (including public displays) through use of visual codes / markers. The codes encoded up to 76 bits of data in a roughly-square configuration and were designed for quick interpretation within the processing constraints of a typical mobile camera phone (at the time a Nokia 7650 running the Symbian operating system). The authors suggest a variety of applications for their visual codes: data could be directly encoded for simple information takeaway or a URL encoded to allow acquisition of larger or more dynamic data, use of URL encoding could also support questionnaire/poll response, whilst taking into account phone orientation, rotation and position relative to the code could allow more complex interaction such as menu selection.

Visual markers were also used by Ballagas et al. in one of two methods for interaction with public displays described in their work "Sweep and Point & Shoot" [Bal05]. The method, Point & Shoot allowed users to select an item at the display by positioning their camera phone to ensure the intended selection area was at the centre of the camera image – at least one marker would also be in view. By calculating the distance between the centre of the image captured on the camera phone from the coordinate represented by the visual marker, the phone could then send an accurate pixel selection to the display. Their second interaction method, "Sweep" allowed the camera phone to be used as an optical mouse. As the user held down their phone joystick and moved their phone in the desired direction, optical flow image processing was used to determine relative motion. All processing was done at the phone to allow multi-user support at the display.

Direct Pointer [JOMS06] also provided cursor manipulation through mobile camera devices. In their system, the mobile camera continuously fed its view of the display back to the screen itself. Each time the mobile device moved, the position of the cursor within its camera frame would change, sending the new cursor position to the large display allowed it to calculate the movement of the mobile device and then move the cursor to correspond with the new centre of the camera frame. An evaluation of the system suggested that interaction with Direct Pointer was comparable to interaction through a trackball or joystick. A similar technique was proposed by Pears et al. [PJO09, POJ08]. Camera views from the mobile device were transferred to the display server over a wireless communication channel to allow the server to map the currently displayed content with the mobile camera view. A prototype system using visual markers was tested by four users suggesting that the approach had potential but significant improvements were needed (i.e. removal of the visual markers, higher frame rates, more robust tracking). A second prototype trialled on higher performance hardware with ten users yielded fewer errors [PJO09]. A mobile camera feed was also used to calculate the position of the mobile device relative to nearby displays in Touch Projector [BBB+10]. Touch Projector allowed interaction between a user and multiple displays in their environment through a mobile device. The camera stream from the mobile device was forwarded to an environment manager that performed operations across the displays (e.g. allowing a user to transfer a content item from one display to another). Touch interactions could also be used to zoom or pause the image at the mobile device – these improvements were found to improve user task performance. A similar technique for display interaction was also used by Gehring et al., but in their case the processing of camera images was done at the smartphone itself rather than requiring the phone to forward frames onto the display [GDL12]. Once the display had been identified, touch interactions were sent to the display using the TUIO [KBBC05, Kal13] protocol.

Shirazi et al.'s Flashlight [SWS09] used a camera attached to a public display in order to track the position of the flash portion of a camera phone. By moving their phone in front of the display a user could control a cursor and zoom the display image; blinking the flashlight on and off could also be used to make a selection. A user study evaluation found the flashlight method to yield comparable task performance to accelerometer-based and mobile phone button-based interaction methods. A similar light-tracking technique was utilised by Miyaoku et al. in their (earlier) work C-Blink [MHT04]. Unlike other systems, C-Blink did not require a camera-phone for interaction but was instead capable of tracking any mobile LCD display. In this approach, a Java application running on the mobile device is used to change the hue of light given off by the device LCD. A camera at the public display detects the signal and calculates the differences in hue as the frames change. Once a valid set of hue-differences are detected they are decoded to give the data transmitted by the mobile device. A prototype system using a large display and mobile phone running C-Blink allowed users to make button clicks, take information away from the display and to transfer images to a specified portion of the display. At the time of the trials only a subset of mobile devices provided sufficiently precise control over their displays to support C-Blink.

In 2010, the use of markers on public displays was revisited by Hyakutake et al. [HOKK10]. The authors placed a polarisation sheet covered in a matrix of 2-D markers (as developed by Koike et al. [KNF09] for use on an LCD tabletop). The markers are largely invisible to the human eye but placing a polarisation sheet over the camera phone lens allows detection at the mobile device. The markers allow the mobile device to obtain its position relative to the display such that the phone can then be used as a pointing device for the display. The authors describe how the system can be used to move and rotate an object shown on a display and the potential for multi-user interaction.

More recently, pico projectors have become viable devices for interacting with (and creating) public displays. Use of such devices to appropriate surfaces to create their own, transient, public displays is described in Section 2.2.4.2. The use of mobile projectors as a method for interacting with public displays was demonstrated in Flashlight Jigsaw [CMB08]. The authors used mobile projection devices to support interaction with a multi-player game on a large projected public display. Pieces of a jigsaw were only revealed when the mobile device projected a ‘flashlight’ over specific portions of the shared display; control of the pieces was also divided amongst the devices, with each device being tied to its own cursor. Users worked together to complete the jigsaws. A two week trial of the system in two locations revealed that users found the system simple to operate. A total of 239 individuals played with the game.

Motion and Haptics Toss-It [YTSH04, YTH$^+$05, YTH$^+$06] provided a mechanism for a user to ‘Toss’ or ‘Swing’ files from a PDA to other electronic devices. For example, in one application scenario, a user transfers a presentation slide from their PDA to a nearby display. The PDA device was fitted with inertial sensors used to detect the direction and force of motion; a Toss-It server stored the position of all local devices such that the target could then be identified. The file was then transferred over wireless LAN. A user study [YTH$^+$06] found success rates of between 55% and 90% depending on the relative position of the target device. As accelerometers have become a common feature of modern smartphones, similar systems have been developed for these platforms. Scheible et al. [SOC08] developed their system, MobiToss, which also allowed users to 'throw' files from a mobile device to a public display through movement of the mobile device. Once the file was received at the display, users could continue to interact with the item by moving their device (e.g. twisting the phone to rotate an image shown at the display). A trial at an academic conference resulted in mostly positive feedback although not all throws were identified.

Holleis et al. [HRKS06] explored the use of 'gestures', movements of a small physical display, in a person-to-place messaging system. An acceleration sensor on the display was used to recognise a limited set of gesture interactions which could be used to reply to an incoming message by selecting from a set of predefined responses. An evaluation with 8 students showed that the gesture set was not intuitive enough for users to determine alone but that it was simple enough to be learnt with a little tuition.

Inspired by the popularity of the Nintendo Wii [Nin13], Vajk et al. [VCBE08] used the mobile phone as a "Wii-Like" controller for their display game TiltRacer. The multi-player game used accelerometer data from the users' mobile phones transferred to a game server over Bluetooth. Tilt actions were used to steer cars around a race track. A trial of the game at two events showed that players found the game fun and intuitive to use. Further projects using mobile phone accelerometers from the same group include Tunnel Run (a first person driving game) and MirageMoney (a flight-simulation game) [Mob13].

Boring et al. used movement of a mobile phone for two of the three interaction methods presented in their 2009 work "Scroll, Tilt or Move It" [BJB09]. 'Moving' used a linear mapping between movement of a mobile phone and movement of a pointer on the public display. 'Tilting' used the joystick of the mobile phone to control the pointer, but allowed acceleration of the movement through the tilting of the mobile device. The final method, 'Scrolling' provided pointer control only through the joystick of the mobile phone. A comparative evaluation of the methods showed that Moving and Tilting could increase interaction speed but also resulted in higher error rates.

Whilst most of the research described in this section has focussed on the mobile device as an input mechanism, and the display as an output device, other configurations are also possible. Rukzio et al. developed the Rotating Compass [RSK05, RMH09], which used the mobile device as a feedback mechanism to highlight user-specific content on a public display. Their application used a rotating compass needle shown on a public display. The display animation cycled through directions in a clockwise manner. To provide navigation information to a mobile user, that user's mobile device would vibrate only as the display highlighted the correct direction for their intended destination. This approach allows the public display to show multiple user's data simultaneously and can be used to provide a degree of privacy by preventing viewers from matching displayed data to other viewers. A user study comparing the Rotating Compass with other navigation methods (paper map, phone only, display only) found that the Rotating Compass and the display-only mechanisms were comparable and that both offered better navigation and usability properties than other methods.

Using Personal Mobile Devices for Information Takeaway

Smart phones can also be used to support 'take-away' functionality that allows viewers to collect information from the display – providing the digital equivalent of the tear-off strips found on analogue posters and flyers. Since the adoption of the camera phone, mobile camera devices have provided a simple mechanism for information takeaway for both analogue and digital displays. In an early study, Okabe cited an example of a user photographing a job advertising poster as a form of "visual note taking" [Oka04, pg5] whilst Kindberg et al.'s study of photos from thirty-four camera phone users found that more than 10% of photos featured an 'image of screen, writing and so on' [KSFS05]. Kindberg also provided an example, a man who took a photograph showing the contents of a whiteboard at the end of a meeting. Digital displays offer many more mechanisms for supporting information takeaway on mobile devices. For example, many of the previously-described mechanisms for display interaction (e.g. visual codes, short-range communication technologies) are also well-suited to information takeaway.

iCapture [MRS06], an application developed as part of Lancaster's eCampus project, focussed on the use of camera phones to takeaway news items shown on displays in transient spaces. News headlines shown on the public displays were each accompanied by a visual code that encoded the URI of the complete news article. When a viewer passing the display found an article of interest they photographed the visual code. The iCapture mobile application then processed the codes, and initiated downloaded of the article over HTTP. Evaluation of the work was done informally through testing by a number of members of the networking team within Lancaster University's Computing Department; comments suggested that the system was simple to use but could be slow to download the requested articles.

Visual codes were also used in Digifieds [AKB+11, ASKS13], a classifieds application for the UBI-hotspot displays in Oulu, Finland. The application allowed viewers to create, read and take away classified adverts through the public display, a web interface and an Android mobile client. Take away was supported through two mechanisms: a user could enter a five-character alphanumeric code into the mobile client that uniquely identified the advert of interest, or they could scan a QR-code shown with the advert. Multiple classifieds could be retrieved simultaneously through use of a 'cart' into which users collected items of interest before scanning a single code that represented the contents. Digifieds was deployed for six months and was evaluated through a combination of observations, interviews, log analysis and a field study. The mobile application was reported to offer some benefits over touch interaction with the display itself – users found interaction with the mobile client to be more responsive than the capacitive displays and could take advantage of other hardware available on their mobile device (e.g. using cameras to photograph an item). Despite low levels of familiarity with QR-codes, most participants quickly understood how to use them and commented positively on use of this method. Strohbach et al. [SKM09b] used QR-codes to allow users to access content related to targeted adverts shown on public displays (as described in Section 2.2.7.3).

Shoot & Copy [BAB+07] allowed the viewer of a display to transfer files from a public display to their mobile device. Each file was represented by an icon on the public display. A viewer used the camera of their mobile device to capture an image of the screen that included the icon representing the file they wished to transfer. The camera image was then sent to the display (via GPRS or Bluetooth) which calculated the area of interest, identified the file within the area, and returned a URL from which the file could be fetched. A usability study with twenty-eight participants found that the average time of 9.2 seconds between capturing the image and receiving the information was acceptable to all users and that participants quickly learnt how to use the system. The system did perform less well in poor lighting conditions.

BlueInfo [KKK+11] allows users to receive information from a public display onto their mobile device using Bluetooth. Following a number of prototypes, the final system was deployed onto the UBI-hotspots. A user could use the touch-screen display to select content of interest. By selecting their mobile device from a list of visible Bluetooth devices, the user could then initiate an OBEX Push connection from the display to their device to transfer the selected content. An evaluation of 100 days usage data showed that over 7000 items were downloaded to ~1300 devices and that news items were the most commonly downloaded.

She et al. [SCFH12]'s SmartSignage system used smartphone accelerometers to detect 'dragging' gestures made using the phone in order to indicate selection of items on a nearby public display. Upon receipt of a dragging event, the public display then broadcasted the requested content item to the smartphone over WiFi. Evaluation showed that the average response time was less than one second and that the method provided scalable and intuitive interaction even in a group setting.

Co-Displays and Cyber-Foraging

Used together, personal mobile devices and public displays can act in a way that allows each to overcome the shortcomings of the other. A mobile user struggling with the limited screen real estate of their device can "forage" for a nearby display and transfer their content for clearer personal viewing or to facilitate shared viewing. Equally, a user struggling to clearly view a detail on a display might use their smartphone as an alternative view or magnifier. Such scenarios have been the focus of a number of recent systems.

Foraging for processing, networking and screen real-estate Perhaps the earliest work that used public displays as a method of providing a screen for small personal mobile devices was The Personal Server [WPD+02]. In this work, a user would carry their personal data on a Personal Server device that provided processing, data storage and short-range (Bluetooth) networking. Following discovery of a host computer, display, or kiosk over Bluetooth, the Personal Server mobile device established a connection that allowed file access, web-service execution, and user interaction through the host.

Pering et al.'s Elope system [PBW05] provided support for mobile devices to forage for displays and other devices within an interactive space. The system was focussed around operations (e.g. transfer a photo album from my phone to this display) which were embedded into RFID tags situated throughout the environment. Upon reading a tag, the mobile device receives the necessary information to form a Bluetooth network, express its intent and complete the operation. Their prototype mobile device was designed to replicate functionality provided by a Nokia NFC mobile phone and their middleware could run on a range of operating systems with support for projected displays.

In 2009, Satyanarayanan et al. outlined a vision for "cloudlet" computing – virtual machines that provided trusted, low-latency, local, cloud functionality for mobile computing [SBCD09]. Wolbach et al.'s [WHCS08] early exploration highlighted the role of communal computers and public displays in cloudlet-based mobile computing. In such approaches, a user's smartphone is used as a transport mechanism for a virtual machine (VM), either by directly storing the files that represent that VM or by carrying a pointer to the actual location of such files. Upon encountering a display, the smartphone triggers the transfer and instantiation of the VM to local computing infrastructure allowing the user to view and interact with their VM at a large public display. Wolbach et al.'s prototype cloudlet platform, Kimberley [WHCS08], demonstrated the viability of transient cloudlet customisation through the application of VM overlays onto a base VM. Demonstration of the system with a varied set of applications showed transmission, creation and startup of the VM could be achieved in approximately 1-1.5 minutes on a 100 Mbps bandwidth network and approximately 3-5 minutes on a 10 Mbps bandwidth network. Their application set was focussed around desktop applications that would, most likely, be unusable on a mobile device alone – foraging for a nearby cloudlet and display would provide significantly improved usability.

Migration techniques have also been demonstrated for web applications. For example, Ghiani et al. present a scenario in which a web application is automatically migrated from a Desktop to a smartphone and then onto a public display [GPP+12]. Whilst in the Kimberley prototype VM instantiation was explicitly user-triggered, in Ghiani et al.'s work the user defined a set of context-dependant rules that triggered automatic migration. For example, a user may define a set of rules that result in their application being migrated from their smartphone to a shared display when their smartphone is stationary and located close that display. Sorensen and Kjeldskov highlight the need for immediacy in automatic application migration, and distributed application, scenarios [SK12]. Like Ghiani et al., they suggest the use of context and proxemics to inform migration decisions.

The use of NFC as a data exchange mechanism for cyber-foraging of displays was explored by Nickelsen et al. [NMS10]. The authors suggested the use of NFC as an ad-hoc communication method for application migration scenarios in which other methods are unavailable or inappropriate. By conducting a set of transfer measurements for application state between 1 and 700 bytes, the authors identify a linear relationship between state size and migration duration and show how this can be used to estimate the maximum state transfer possible during the limited time a user might be expected to hold their mobile device in range of the display. For example, in a five-second period they estimate that an application state of 2197 bytes could be transferred to the display.

Commercially, support for cyber-foraging from mobile devices, is perhaps best-represented by wireless screencasting standards such as Miracast [Wi-12].

Not all cyber-foraging research combining mobile devices and public displays has focussed on the individual. Leikas et al. [LSI+06] explored the use of a public display as an method of bringing additional value to a mobile multi-player game. A user trial with thirty-two participants compared gameplay on the mobile phone alone, with the same game played on the phone and shown at a public display (interaction with the game was through the mobile phone only in both conditions). The addition of the public display increased communication between the players and resulted in added value as players felt they were able to get to know each other better.

Use of the mobile device as a private co-display The projects described above have centred around methods by which the display can extend the capabilities of a mobile device. By contrast, the following paragraphs describe systems that typically used the mobile device as a method of augmenting functionality provided by a public display.

The use of mobile devices to offer a personal view of public displays provides a useful mechanism for supporting personal content in a privacy-preserving manner. Shoemaker and Inkpen coined the term 'Single Display Privacyware' [SI01] to refer to a technique in which private information is shown on the public display but is filtered or transformed to prevent viewing. A user can then view their private data on the display with the assistance of a personal mobile or wearable device. Their prototype system used a modified active shutter 3D system to allow two users to view personal data on a public display. Two pairs of glasses synchronised frame transitions with the display. Public information was shown during every image frame whilst private frames were divided: one user's private data was contained within odd-numbered frames and the other within even-numbered frames; the synchronised glasses ensured the user only saw their own frame set by closing the lenses for every other frame [SI01]. Berger et al. [BKN05b] used blurring to obscure private data on a public display; an interface on the user's personal mobile device allows them to select a blurred portion and view the original data on the mobile device. In their prototype, a smartwatch was used to discover a local display for display of an email message, choose the level of data sensitivity (i.e. the proportion of text to be blurred), and to select and reveal a blurred word. Blurring was also used as a technique for masking private data in a prototype developed by Sharp et al. [SSB06]. A VNC connection from a PDA device allowed a user to mouseover blurred portions to reveal them. No evaluation was presented for either Berger et al. or Shoemaker and Inkpen's prototype. Sharp et al. evaluated their text removal process, finding a typical success rate of almost 100%, and conducted a usability study in which eight participants successfully found information from email messages and word processing documents [SSB06].

Alternatively, the use of mobile devices as private co-displays can simply offer the opportunity to provide a clearer or more detailed view of a content item shown on the display. Such views may allow a user to improve a poor viewing angle, to see past an obstruction that is blocking the display, or to enhance a small detail. Lee et al. describe the use of smartphones to provide an ‘individual view’ [LKS+11] of a shared display in a public space. Their system, S^{2}Interaction [LKS+12], uses an Android smartphone to render a specific portion of the content shown on a large display. Movement of the smartphone is tracked using a Kinect [Mic12] and the view onto the display content is updated accordingly. A View Manager component maintains a list of all devices’ local and relative positions to allow multiple users to view the display simultaneously. Interaction with the display is also possible through the touch-screen of the smartphone (touch events are forwarded to the View Manager). The use of individual views of a public display shown on a mobile device was also explored by Cheng et al. [CLMT12]. Their work used tablet computers to allow a user to have a view of the whole display content (a ‘world-in-miniature’ view) and to also be able to view portions of the display in greater detail (a ‘focus view’). Selection of the focus view is achieved through the world-in-miniature view using touch interaction. Communication with the large display was achieved using VNC. Like Lee et al.’s work, Cheng et al.’s prototype system allowed multiple users to view and interact with the larger display simultaneously through their mobile device; no evaluation was presented for either system.

2.2.7.3 Advertising and Novel Business Models

In the following sections we explore a number of works focussed on commercial concerns – first we focus on the motivation for display installation in commercial spaces, second we detail research that has explicitly focussed on the use of pervasive displays as an advertising medium, and finally we summarise activities that have explored methods for selecting the optimum advertisement to show on a display.

Motivating Commercial Display Use

Müller et al. [MWP10] visited 415 stores on eight shopping streets in Münster, Germany. Fifty-three of the stores (approximately 13%) were found to have some kind of digital signage. Each owner was asked why they had installed digital signage in their store; a total of ninety-nine reasons were elicited with a typical response including one or two reasons. A follow-up question about how the owner had become aware of digital signage elicited an average of one answer per store, for a total of fifty reasons. Two independent raters categorised the answers using an affinity method producing twenty-one distinct reasons for installing the signage and seven distinct comments regarding awareness. Key motivations for shop owners to install signage included: the idea that attention is attracted by animation (sixteen answers) or by the displays themselves (seven answers); the ability to advertise products from themselves or other companies to promote sales or bring in additional income (eight answers); that they allow more content to be shown (six answers) or allow the content to be changed quickly (eight answers); or that the displays were mandated by a store chain's headquarters (seven answers). Direction from a central store headquarters was usually the primary method by which the store owners had become aware of digital signage (thirty- four answers).

Pervasive Displays as an Advertising Medium

Ranganathan and Campbell [RC02] highlighted the potential for public displays to act as a medium for pervasive advertising. Commercial deployments of displays purely for advertising are now commonplace (as described in Section 2.2.7.1) and advertising often also features heavily even on 'informative' displays.

Rakkolainen and Lugmayr [RL07] demonstrated use of their FogScreen (previously described in Section 2.2.4.1) as a novel medium for walk-through advertisements. A laser scanner provided walk-through detection allowing advertising campaigns to provide users with a coupon for passing through the advert.

Valkama and Ojala [VO11] interviewed stakeholders of the UBI-hotspot deployment in Oulu, Finland (see Section 2.2.7.1). Existing UBI-hotspot advertisers reported that their customer feedback was positive, this was supported by survey results from 266 city citizens in which almost seventy percent reported that advertisements on the UBI-hotspots increased positivity for the advertiser. Existing advertisers also found the pricing reasonable. By contrast companies who had not previously used the UBI-channel were unlikely to express agreement with the statement "UBI-channel is a good way to advertise" with 23% blaming their disagreement on the pricing of the UBI-channel.

Alt and Schneegass [AS12] focussed on the overall architecture of a pervasive display system with support for advertising alongside other content. Their work presents three architectural variations: one advertiser-centred, one centred around the viewer/user, and one hybrid 'trusted' architecture. Each of the architectures include five key components: a *display client* including display output, sensor input and interaction support; a *scheduler* to determine the presentation of content onto a display; an *application store* to facilitate the acquisition of applications at a display; a selection of *content*, advertising and non-advertising items, which may display within an application or standalone; and a *log* containing execution and interaction data for applications and content. In their three architectures, the placement of each component differs in order to best meet a particular set of stakeholder requirements. For example, in the advertiser-centred architecture, the log, scheduler and content provision are all within the advertiser's domain whilst in the user-centred architecture the log is held by the user, on their mobile device to ensure privacy. In the trusted-architecture an attempt is made to balance the advertisers need for scheduling certainties and data against the viewer's interest in seeing varied content and in maintaining privacy; in this architecture both the scheduler and log are held by a trusted-entity.

Alt et al. provide an overview of issues related to advertising on public display networks [AMS12]. They identify three methods for integrating advertising with other forms of content: time-multiplexing (cycling through content such that within a set time period both advertising and non-advertising content has been shown), space-multiplexing (physically dividing available screen space such that both advertising and non-advertising content can be shown simultaneously), and integration (integrating logos and other advertisement features into non-advertising content). They highlight the need to create advertising content that specifically addresses the challenges and opportunities of a public display environment (e.g. divided attention, potential for highly- dynamic engaging experiences). They also identify some factors that may be important in the success of advertising on pervasive display systems: understanding audience behaviours, measuring performance, attracting attention and interaction.

Optimising Advertisement Selection

Many researchers have explored the problem of advertising selection, and methods for optimising advertising strategies on pervasive displays. Payne et al. [PDJS06] proposed the use of auctions to select advertisements on public displays. Their BluScreen display used Bluetooth scanning to count users in front of the display and build up user history records. A repetitive second-price sealed bid auction was used to determine the winner from a set of advertising agents, each placing a bid on behalf of an advertisement to be shown.

Müller and Krüger [MK07a, MK07b, MSK07] explored the use of auctions for selection of 'actionable advertisements' as part of their work on multicast communication via public displays [MK06]. By building user profiles (like Payne et al., this work also used Bluetooth sensors) their systems could calculate the probability that showing an advertisement would result in the desired action, informing auction bidding. Using data from previous actions also allowed their system to infer user interests and therefore integrate recommendation algorithms into their bidding agents [MK07b].

The use of user profiling for public display advertisements was explored further by Alt et al. [ABK+09]. They created a system in which user preferences and behaviour (detected through use the user's mobile device) were combined to create a user profile. To select appropriate advertisements at a display the system either selected the optimal advert for each viewer in turn, or selected adverts that were a good fit for the majority of viewers. The work was deployed in a laboratory setting and no evaluation was reported.

Müller and Krüger [MK09] deployed a prototype coupon-system, MobiDiC, on twenty 13" advertising signs on public telephones in Münster, Germany in September 2007. The system was intended to build on their prior research into context adaptive scheduling [MKK07] in which extrapolation from previous behaviour patterns (in this case, coupon redemption) would allow the displays to calculate a utility value for available advertisements and schedule only those with the highest utility values. Their twelve-month deployment involved coupons for seventeen stores but only thirty-seven coupon redemptions occurred (these coupon redemptions were also clustered over only five unique coupons – the most popular was redeemed seventeen times). The system also suffered from novelty effects

– almost 90% of coupon redemptions occurred in the first three months of the deployment. Due to the low number of coupon redemptions, the system never reached a point at which it had sufficient data to move from random to adaptive scheduling.

Strohbach et al. [SKM09b, SKM09a] created a prototype system using Bluetooth tracking to identify participating users. Their system, built using the Context Management Framework (CMF) [SLO$^+$06], uses a Bluetooth ping technique (cf. l2ping [Yev11]) to detect the presence of registered devices. Users could register their interests alongside their Bluetooth MAC address and content could then be targeted towards their interests. Using a QR-code displayed as part of each advertisement, users could view additional content and a barcode coupon on their mobile device. The system was demonstrated over three days of a trade fair but no formal evaluation was reported.

2.3 Commercial Support for Digital Signage

Beyond the research domain, displays in public and semi-public spaces are now commonplace. Deployments vary in size and purpose. Larger-scale commercial deployments (1000s of displays) are typically owned by advertising brokers. Notable deployments of this kind include the INFOSCREEN networks in Germany, Austria and Malaysia (as described in Section 2.2.7.1), the Adspace Digital Mall Network [Ads14], the JCDecaux out-of-home advertising screens [JCD14], and the GSTV [Des14] displays embedded in petrol station pumps across the United States. Other commercial display deployments are typically acting in a digital signage role, and a wealth of commercial-grade software exists to support this functionality. In this section we provide an overview of established signage platforms.

Scala [Sca14] provide tools for content design, management and playback; their content management service can also be accessed as Software as a Service. The Scala content player supports a wide range of media types including audio-only, media streams, and interactive (touch-screen) content; the player also continues to play when disconnected from a network. The Scala content manager supports playlist generation, remote player monitoring and provision for emergency alerts, The Sony Ziris suite of applications [Son14] provides a similar set of tools, with Ziris Create supporting content creation and management, Ziris Edge providing content distribution, Ziris Manage allowing display management, and Ziris View providing media playback. In addition, more recent versions of Ziris support a variety of video wall configurations including displays at a variety of angles in different sizes.

BroadSign [Bro14] is a Software as a Service platform for digital signage that includes remote scheduling and administration for signage networks, file storage, a caching engine and media player. JCDecaux [JCD14] has used BroadSign for its new installations since 2011. Like Scala, the BroadSign player can cope with periods of disconnection. Support for advertising content is provided in the form of proof-of-play accountability, audience measurement, targeted advertising (based on audience, weather, location etc.), and competitive separation (i.e. not playing similar adverts together). Similar provision for targeting scheduling around sales is provided by Dynamax's POV^{NG} system [DYN14]. An alternative area of focus can be seen in Haivision's CoolSign [Hai14], which provided integration of digital signage functionality with IP Video content.

In contrast to the previous systems, Appliance Studio's PrintSign [App14c] provides lightweight signage support. The PrintSign allows content creators to 'print' media to physical signage through a special network print driver that incorporates scheduling criteria. Other lightweight digital signage approaches typically use a web browser as the signage player, with logic for content management and scheduling offloaded to a web server (e.g. Zetakey [Zet14], Beabloo [BEA14]).

Open source software for digital signage may also be used in commercial settings. Xibo [GHtXP14] provides an open source server application for content management and scheduling control, together with open Windows, Linux and Android clients to playback content and generate simple analytics. Support for the open standard, SMIL [W3C03] (the Synchronized Multimedia Integration Language) is a common feature of both open and commercial signage software (supported, for example, by Xibo, Scala and Coolsign amongst others). SMIL [W3C03] is a World Wide Web Consortium recommended XML-based document format designed to describe multimedia presentations. The format features markup to describe presentation elements such as timing, layout, transition animations together with media elements such as text, video, audio and images. Each SMIL file can link to other SMIL files (e.g. to nest one presentation inside another) and support is provided for using interaction events (button presses, motion events) to trigger specific sets of content. Although originally focussed on presentations, advocates have highlighted the suitability of SMIL as an open standard for digital signage systems.

2.4 Understanding Viewer Behaviour

Understanding how viewers respond to displays is an important research topic. In this section we provide a survey of work in this area including models of how people move around a display (audience flow models), understanding viewer engagement levels and the factors that affect the degree to which viewers engage with displays, and experiences of engaging multiple viewers at a display simultaneously.

2.4.1 Audience Flow Models

Audience flow models provide methods for describing the spaces around a display, typical viewer actions in those spaces, and the transitions from one space to another as viewers move around a display. Understanding this behaviour is particularly important when designing interactive displays or displays that change their content based on the presence of one or more viewers.

System-Driven Models
In their research with the Hello.Wall (previously described in Section 2.2.4.1), Prante et al. [PRS+03, SRP+03] divided the area around the display into three zones [Figure 2.1a]. Two RFID readers with different ranges provided the bounds for the zones. The area furthest from the display and therefore out of range of both readers was referred to as the *Ambient Zone*. The area around the display that was close enough to be in range of the long-range RFID reader but not close enough for the short-range reader formed the *Notification Zone*. Finally, the area close to the display in range of both readers was termed the *Interaction Zone*. The

authors used their three zones as a mechanism for supporting "distant-dependant semantics", in which the distance of an viewer from the display determined its behaviour. In ambient mode, the Hello.Wall showed general information. As an individual approached the display (and entered the Notification Zone), the display began to show user-specific notifications and provided limited interaction support. Moving much closer to the display (into the Interaction Zone) allowed a greater degree of interactivity.

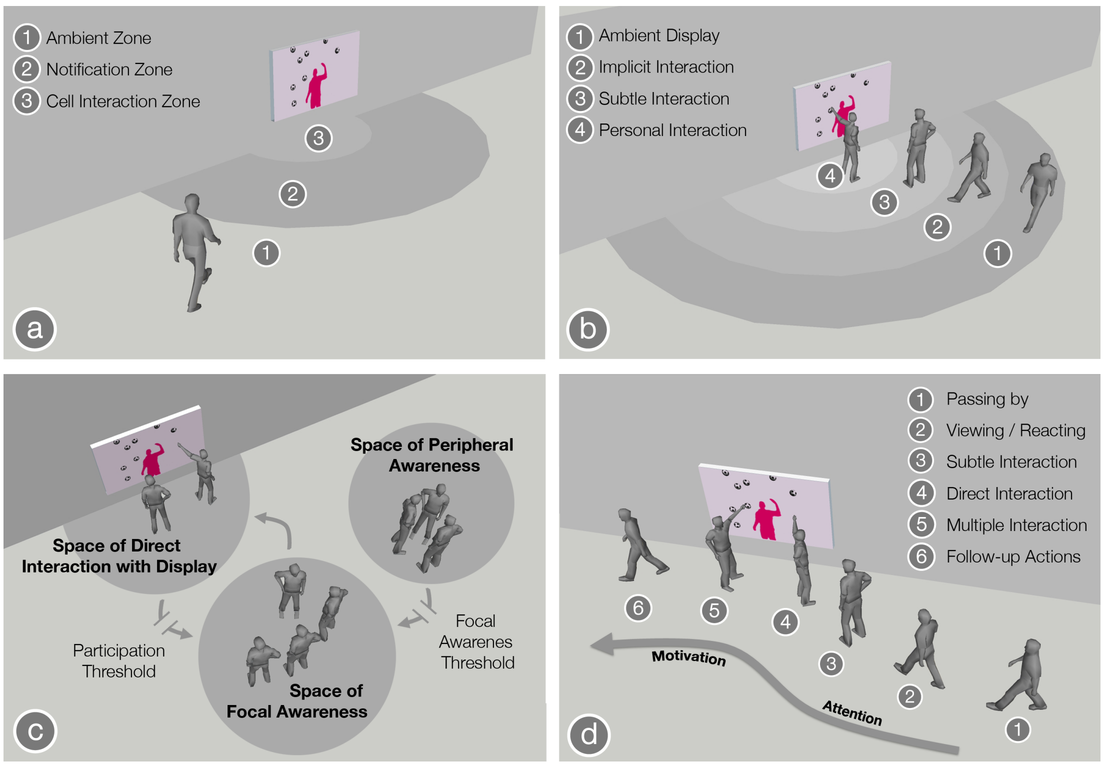


Figure 2.1: Interaction Models for Public Displays: **(a)** Hello.Wall Zones of Interaction [PRS+03, SRP+03], **(b)** Vogel and Balakrishnan's Interaction Framework [VB04], **(c)** Public Interaction Flow Model [BR03], **(d)** The Audience Funnel [MM11].
Image © Florian Alt. Used with permission.

Hello.Wall's interaction zones were used as a basis for Vogel and Balakrishnan's interaction framework [VB04]. Movement through their four phases (shown in Figure 2.1b) was determined by combining proximity (as used in Hello.Wall) with additional cues such as body orientation and gestures. The framework featured an *Ambient Display* phase with slow- updates and an undetailed information overview; an *Implicit Interaction* Phase, triggered as a user passes the display, that was used to display important notifications and invite the user to further engage with the display; a *Subtle Interaction* Phase with detailed information display, display of personal (but not private) data, and support for gesture interactions; and a *Personal Interaction Phase*, in which touch interactions were supported and limited private information could be displayed (occluded by the physical presence of the user). As viewers moved through the phases to engage with the display they would spend increasingly large amounts of time in each phase (e.g. glancing in the Ambient Display Phase versus 2-5 minutes for the Personal Interaction Phase) allowing them to consume increasingly detailed information and participate in increasingly explicit interaction sequences (see also Section 2.2.4.2).

Observation-Based Models
As a result of observation studies of behaviour around their display, The Opinionizer (described in Section 2.2.6), Brignull and Rogers [BR03] identified three activity spaces and highlighted the importance of the thresholds (transition points) between them [Figure 2.1c]. In a *Peripheral Awareness* space people are engaged in activities that do not involve the nearby display. People in the *Focal Awareness* space are engaged in social activities around the display (e.g. observing others' interactions, discussing content on the display). Finally, the *Direct Interaction* space allows users to actively interact with the display itself. The authors highlight the importance of display position in encouraging users to cross the Focal Awareness Threshold (i.e. to move from Peripheral to Focal Awareness) and for straightforward interaction that can be learnt through observing others to encourage users to cross the Participation Threshold.

Michelis and Müller's Audience Funnel [MM11] followed an observation of four adjacent 'Magical Mirror' outdoor, urban, displays installed in Berlin, Germany. They identified six phases of user interaction for the Magical Mirrors installation [Figure 2.1d]. The *Passing By* phase described the state of potential viewers (i.e. those who could see the display from their current position but are not currently looking). Once a passer-by engaged in some observable reaction to the display they were said to have entered the *Viewing and Reacting* phase.

Deliberate movement to trigger a response at the display (i.e. interaction) was divided into three phases: *Subtle Interaction*(for initial, brief, movements some distance from the display); *Direct Interaction* (for very deliberate movements closer to the display); and *Multiple Interaction* (for cases in which a single user interacts with more than one display or interacts with a single display more than once within a short period). Finally, a user sometimes entered a *Follow-Up Action* phase, performing some other activity related to the display (e.g. taking a photograph of the display). The authors identified that movement through the phases may not be sequential (e.g. a returning user may skip some of the early phases) and that not all phases are applicable to all display deployments (e.g. the Multiple Interaction phase would be less applicable in single-user deployments). A second observation period was used to explore thresholds between the phases. The most difficult threshold to cross appeared to be the transition from passing-by to subtle interaction (only 33% of passers-by made this transition). Once a user entered into any form of interaction, the likelihood that they would progress into other phases was considerably higher (e.g. 95% of those in the Subtle Interaction phase would move into a Direct Interaction phase).

A number of other researchers have provided more coarse-grained categories for observing those around public displays. For example Tang et al. identified three classes of individual around their interactive MAGICBoard displays [TFB+08]: *passers-by* (individuals in transit who may glance at the display for no more than ten seconds), *standers-by* (those stationary around the display whose primary goal is not display-related but whose location may allow them more time to view content) and *engaged bystanders* (those who are actively making use of the display). Finke et al. [FTLB08] identify *Bystanders*, *Spectators* and *Actors*. Whilst Hardy et al. distinguish between those who *ignore* a display, those who *glance* (<3s) and those who *watch* [HRD11].

2.4.2 Viewer Engagement

Observational studies have provided significant insight into current low levels of viewer engagement (Display Blindness) and have identified three key challenges to increasing this engagement: attracting attention, communicating interactivity and motivating interaction.

2.4.2.1 Display Blindness

In 2008, Huang et al. [HKB08] conducted a set of real-world observations around public display deployments in three European cities. They observed forty- six displays, mostly flat panels but some projected, of which only two were interactive. Each display was observed over a series of visits (on differing days /times) for at least one hour per visit. Typical content items shown at the displays included short cycles through a set of still images and simple animations (e.g. text transitioning onto the screen). Seven of the displays (~15%) showed only a single still image. Huang et al. found that passers-by paid very little attention to the displays; glances were infrequent and typically limited to one or two seconds (maximum ~8 seconds). A number of factors were found to affect the level of attention received by a display: those positioned at eye-level received a larger number of glances than those mounted very high or very low; video content was more likely to receive longer-lived engagement; displays close to a physical artefact were likely to benefit from attention transfer from the artefact to the display; finally, smaller displays were more likely to receive prolonged viewing, particularly if those displays were also interactive (no large interactive displays were available for comparison). A number of factors were also identified as having little or no impact on the frequency or length of glances: there was no evidence that particular content types (e.g. art, advertisement, news etc.) were more likely to attract attention; nor was there any evidence that attention was more frequent or more lengthy in 'captive audience' situations (e.g. queues).

The term 'Display Blindness' was coined by Müller et al. [MWE+09] who interviewed passers-by around a sample of eleven displays in Münster, Germany. Unlike Huang et al. [HKB08], the authors of this study did find that a captive audience were more likely to view the displays. Overall they found that the expectations of passers-by was a highly significant factor in determining whether they would attend to the display. 96% of participants reported that they had looked at the screen in locations with high-levels of familiarity with the displays (the university and civil offices); in these locations participants reported that they were familiar with the content (96%) and that the content was interesting to them (83%). Less than 1% of participants reported looking at the display in locations where the content was unknown (e.g. public telephones, cafés, banks, shops and malls) and expected to be uninteresting (e.g. advertising).

In a second study, Müller et al. [MWE+09] collected ninety-three videos featuring in-situ displays. They presented seventeen participants with a random subsample of ten videos for classification using the repertory grid technique. Once new dimensions had stopped emerging (typically after ten triples), participants were asked to rate the videos on a file dimension reflecting whether they would look at the display. Elicited dimensions were then categorised by two independent raters using the bootstrapping technique. A total of twelve categories were identified and correlated with the final dimension (how likely they would be look at this display in this situation). The highest correlation rates were found for content aesthetics (colourful content 57%, attractive content 43%, content visible from a distance 40%) and the message conveyed by the content (interesting content 54%, emotionally appealing content 40%). The correlation between watching the display and having time to wait near the display (i.e. the captive audience effect) was found to be 36%. None of the elicited dimensions explored the impact of the physical positioning of the screen highlighted by Huang et al..

A study by Hardy et al. [HRD11] also found evidence of Display Blindness. Their study of an interactive display situated on a university campus found that 84% of passers-by completely ignored the display, 15% were seen to glance at the display and only 1% stopped to watch it. However, where an individual passed-by the display close enough to trigger a transition of the screen into interactive mode, that person was also found to be significantly more likely to attend to the display. A number of factors were found to have no significant impact on attention levels: gender and age were insignificant, as were time of day and social setting. Content type (weather forecast or asteroids game) was found to have little impact on the number of glances, but the game did attract significantly more longer length (>3s) observations and the only uninvited interaction attempt. Unlike previous studies, the authors noted that in many cases passers-by were engaged in other activities that would be likely to inhibit interaction with the display although this did not always prevent engagement (the one interacting user was talking on their phone at the time).

During a study of their Looking Glass system, Müller et al. [MWB+12] observed that display blindness was the norm for individuals passing a display alone but was considerably less likely when groups passed the display. They reported that individuals were more likely to pass the display looking ahead and downwards whilst walking faster than groups. When interviewed, individuals rarely reported noticing the display and not one had observed that the screen was interactive.

By contrast, Michelis and Müller's observations of the Magical Mirrors installation noted that "only very few passers-by hurried past the displays without noticing them" [MM11, preprint p. 11]. In both observation phases, the authors noted that the majority of passers-by did glance at the displays (e.g. approximately 650-650 of the 660 observed in the second phase) but that a much smaller proportion (approximately one third) actively engaged with the display. Whilst the authors do not attempt to explore the cause of their high viewing rates, they do note that the glances often followed a reaction from the display to the presence of the passer-by. This echoes the results found in Hardy et al.'s study.

Display blindness continues to be observed in deployment studies (e.g. Goncalves et al. [GFH+13]). Kukka et al. [KOK+13] have also reported observations of explicit 'display avoidance' in which individuals actively turn their head away from a display. Interviews with passers-by who engaged in display avoidance suggested that such behaviour may be a product of their ubiquity and the resulting information overload – "I never look at them anymore... [its] just too much to handle" [KOK+13, p. 6].

2.4.2.2 Attracting Attention

In order to overcome display blindness, a number of methods have been used to draw attention to public displays. Using animation at the display is a common technique, often seen in the form of attract sequences or advancing images designed to trigger a behavioural urgency response. Observation studies have also noted a honey-pot effect in which groups of users around a display attract attention.

Attract Sequences

An attract sequence typically takes the form of a slideshow or animation that shows the potential of the display. Attract sequences may also be combined with calls to action – graphical or textual cues that are intended to explicitly inform a user that a display is interactive (see Section 2.4.2.3).

Flashlight Jigsaw (see also Section 2.2.7.2) attempted to entice passers-by into playing through an animated sequence in which a flashlight randomly scanned through the display area revealing puzzle pieces [CMB08]. An attract sequence was also used in Polar Defence [FTLB08] (see also Section 2.2.7.2). 70% of the screen was given over to an animation that illustrated gameplay. The authors report that the animation acted as an eye-catcher and that they frequently saw passers-by stopping and glancing at the display. The sequence also served as a learning tool, simulating interaction and gameplay for others to observe.

Peltonen et al. [PKS+08]'s experiences with the CityWall suggest that attract sequences may still go unnoticed. They highlight an observation of a group of people sheltering from the rain next to the display. Despite constantly moving objects on the screen, they fail to notice it until others approach. A number of factors can impact the effectiveness of attract sequences. Churchill et al. note that whilst viewers respond positively to changing content, the rate of transition is important; whilst slow transitions appear static and uninteresting, overly quick changes are disturbing [CGNL04]. Ju et al. conducted a study with both on-screen and physical attract loops, finding that the physical objects were more effective at enticing passers-by to interact.

Attract sequences may also be reactive. In Beyer et al. [BAM+11]'s framework for cylindrical displays, they created an application 'tales around the column' that presented an initial set of images that followed the location and trajectory of a passer-by. To complete the image set, the viewer had to continue walking around the display, completely circling the cylinder.

Behavioural Urgency

Franconeri and Simons [FS03] reported that looming objects and those that were moving towards the user would capture their attention. They hypothesised that such objects received attention because they potentially indicate a need for immediate attention.

Within the public display domain, Beyer et al. [BAM+11] developed an application framework for flat and cylindrical displays that allowed them to create content items that fly towards the user. Camera sensors tracked the position of the user. Whilst the feature was not used for their evaluation study, the authors suggest that it could be useful for presenting content that the user "cannot disregard"[BAM+11, p. 4].

The Honey-Pot Effect
As part of their work on The Opinionizer (described in Section 2.2.6), Brignull and Rogers [BR03] noted that as the number of people around the display increased, this in turn attracted the attention of others leading them to discuss, observe, congregate around, and attend to, the display. They dubbed this the 'honey-pot' effect. The honey-pot effect was observed in two separate trial settings: a book launch event at a large academic conference and a, much smaller, welcome party for new postgraduate students at a UK university.

Further evidence for the existence of a honey-pot effect has been reported in other studies. Observations of the MAGICBoard [TFB+08] (reported in Section 2.2.7.2) highlighted the propensity of groups to gather around an interactive kiosk-style display, particularly when contrasted against a display whose interaction was achieved through a personal mobile device. In their work on Flashlight Jigsaw (described in Section 2.2.7.2), Cao et al. report that individuals who passed near the display for some other purpose (e.g. to fetch printouts or coffee) would get sidetracked by the display if others were already interacting. Observations of the CityWall installation in Helsinki, Finland (see also Section 2.2.7.1) found that people "most often notice the wall when someone else is using it" [PKS+08, p. 5]. In 19% of cases the display was already in use when a new user arrived and began interacting and the authors also note that in some cases those who had been stationary near the display and had not previously noticed it suddenly began attending to the display once others had arrived at the display and begun interacting. Similarly, Hardy et al. [HRD11] noted that despite typically low levels of interest in their gesture-enabled display, passers-by would often stop walking past and watch the display if someone else was interacting. Finally, observations of the Magical Mirrors and Looking Glass installations in Berlin, Germany found that evidence of a honey-pot effect: "whenever there was already somebody interacting with the display, it was much more probably that somebody walking down the sidewalk would also start interacting" [MM11, preprint p. 14].

Finally, Goncalves et al. [GFH+13] also note the existence of a honeypot effect, in the form of 'attractors', individual users of a display who attract others to join them and assist in the completion of tasks. However they note that the resulting disturbance often had a negative affect on tasks being completed at the display.

2.4.2.3 Communicating Interactivity

Whilst observations of Display Blindness typically suggest that many people fail to notice the public displays in their environment, many studies also note that even those who clearly gaze at a display most often continue past the display without stopping to interact (e.g. [TFB+08]). Although interactive screens are becoming increasingly common in a range of public spaces, the dominance of non-interactive displays means that passers-by often fail to identify interactive displays. This phenomenon was identified as 'interaction blindness' by Ojala et al. [OKK+12] when they observed the effect on their UBI-hotspot displays in Oulu, Finland (see also Section 2.2.7.1). Interviews and diary studies of their displays showed that the effect was seen across all population demographics. The authors suggested that the lack of a visible input device for touchable displays was a particular problem. Hardy et al. [HRD11] also note the lack of expectation for interactivity, highlighting the case of one group of men who assumed that the interactive game was a simulation, controlled by the experimenter.

Common methods for communicating interactivity include: explicit calls to action on the display or within the nearby environment; reactivity; environmental design to create landing zones; and use of other individuals to provide instruction, demonstration or invitation to interact.

Calls to Action

A call to action usually takes the form of a textual instruction shown on (or nearby) a public display. For example, during periods of inactivity Jacucci et al.'s Worlds of Interaction showed 'help spheres' that emitted slogans designed to encourage people to interact [JMR+10] and Kules et al. [KKP+04] annotated portions of a conference photo display with interaction instructions after sixty seconds of inactivity. Iconic representations may also be used, sometimes as part of an animated attract sequence (see Section 2.4.2.2).

However, evidence from the UBI-hotspot displays (described in Section 2.2.7.1) suggests that such calls to action may not be effective in long-term deployments. Studies showed that use of a 'touch me' animation, which appeared when individuals were close to the display, had no noticeable effect on user interaction [OKK+12].

Kukka et al. attempt to identify visual factors that impact the effectiveness of a call to action [KOK+13]. Their study explored the effect of three visual elements (colour, graphics, and animation) on enticing interaction. A total of eight element combinations (signals) were created: coloured animated text, coloured animated icon, coloured static text, coloured static icon, greyscale animated text, greyscale animated icon, greyscale static text and greyscale static icon. The words "touch me" were always used as the textual representation, and the icon of a hand with a finger extended into a small circle was selected as the graphical representation. For coloured signals, yellow foreground was placed over a blue background; greyscale signals used white on grey. The signals were trialled on eight displays across the University of Oulu, Finland. Each display ran a different signal each day, and the study ran for eight days until each signal had been shown for a day at all locations. A total of 1863 touch-interactions were elicited. The authors report that overall signals with text were more effective at triggering interaction than those with an icon; coloured animated text attracted the largest number of interactions (301) followed closely by the other text signals (277-299 interactions each); iconic representation received approximately 100 fewer interactions (134-197). The role of colour and animation was more complicated: for coloured text, an animated signal was more effective than a static one, but the opposite was true for greyscale text and both coloured and greyscale icons.

Reactivity

In Section 2.4.2.1 we reported Hardy's observation that when a user passed close enough to the display to trigger a transition of the screen content, that person was also found to be significantly more likely to attend to the display [HRD11]. Such reactivity may be used to trigger attract sequences or calls to action (as in the UBI-hotspot deployment [OKK+12]). Reflecting the viewer's own physical movements back to them may be a particularly useful display reaction for communicating interactivity [MM11, MWB+12].

In their work, the Looking Glass [MWB+12], Müller et al. explored methods of providing feedback on audience movements in order to communicate interactivity. The authors compared reactive sequences with and without user representations with more traditional attract loops and calls to action. In the reactive case, passers-by would see either a mirrored user image or silhouette. Similar non-reactive representations were also used in the attract sequences. An observation study found that the reactive representations resulted in significantly more interactions than traditional calls to action. Representation type (text, image or silhouette) was not found to be a significant factor for traditional calls to action but was significant for the reactive sequences (a reactive silhouette resulted in 47% more interactions than the call to action, whilst a reactive user image resulted in 90% more interactions).

Landing Zones

Hardy et al. [HRD11] and Muller et al. [MWB+12] have highlighted the relatively short window of opportunity associated with the time people pass by an public display. In their observations of a reactive outdoor display, Muller et al. note that when individuals pass a display that is not currently being interacted with (i.e. no honey-pot effect), "passers-by often recognize interactivity after they [have] already passed" [MWB+12, p. 2] – to interact with the display they then have to walk back. Muller et al. term this the 'Landing Effect'.

Michelis and Müller's Magical Mirror displays allowed users time to perceive the interactivity of the display through creation of a 'landing zone' [MM11]. Four displays were placed in sequence in adjacent shop windows. The authors reported that users would stop to interact with the second, third or fourth displays after noticing a reaction at the first ones they had passed. After some time interacting with a later screen, passers-by also often returned to an earlier display to explore the interaction at that display as well.

Explainers, Verbal Invitations and Social Learning
In many short-term trials, researchers have been used as a method of explaining, inviting or demonstrating use of a public display (e.g. [CMB08]). Such techniques are not typically seen in longer-term deployments – a rare exception are Oulu's "UBI guides", students paid to encourage and instruct the general public in use of UBI-hotspots [VO11].

For multi-user display systems, invitations from those already using the system may encourage others to interaction. For example, in studies of their system Flashlight Jigsaw, Cao et al. [CMB08] note that twelve of their twenty-seven interviewees reported having invited others to play the game. Observations showed that invitations were common, sometimes addressed at specific individuals, and at other times just generally to those within hearing distance. Invitations made by players were more successful in recruiting others than invitations made by the game facilitator.

Interacting users may also offer instruction in how to interact with a display, either implicitly through demonstration and social learning, or explicitly with deliberate teaching behaviour. Kules et al. [KKP+04] noted that during an early trial of their conference photo display, users typically learnt how to use the system by watching another user; such observations typically lasted between 15 and 60 seconds. Similarly, with their games Flashlight Jigsaw [CMB08] and Polar Defence [FTLB08], researchers also noted that new players were frequently seen to observe gameplay before joining in themselves. Jacucci et al. also observed social learning with their system Worlds of Information [JMR+10]. Users frequently reported having learnt how to interact with the system by observing others and then imitating their behaviour; others also reported having worked together as a group to try and understand the system. Social learning was also reported in early interactive display studies such as BlueBoard [RDS02, RTD04].

In many of these systems, more explicit instruction from experienced to novice users was also observed. For example, Finke et al. [FTLB08] frequently observed people issuing others with instruction for their display application, Polar Defence and Cao et al. [CMB08] noted that experienced players spontaneously tried to teach their game (Flashlight Jigsaw) to others, helping to lower the entry barrier for the game and creating a welcoming atmosphere. They also report that players found the teaching behaviour a rewarding experience in itself. Peltonen et al. also report on cases of individuals learning from, and instructing, others during interactions with their large interactive display CityWall [PKS+08] and Müller et al. report on an occurrence of an apparent stranger approaching an individual by the display in order to show them how the interaction worked [MWB+12]. Similarly, in their study of displays at the University of Oulu, Goncalves et al. [GFH+13] use the term 'herders' to describe individuals who appeared to direct small groups of friends to a display in order to to explain or demonstrate interaction; herding behaviour accounted for approximately 6% of interactions seen in video of the displays.

2.4.2.4 Motivating Interaction

Once a viewer understands that a system is interactive, the final significant hurdle to overcome is motivating the user to actually engage in interaction.

Cao et al. [CMB08] interviewed players of their game Flashlight Jigsaw (see Section 2.2.7.2) in order to identify factors that motivated individuals to interact with the game. Five such factors were identified: novelty and curiosity (17/27 study participants); entertainment and a sense of accomplishment from beating other players (17/27); the influence of other players (e.g. direct invitations from others, noticing other players and the social benefits of playing the game – 16/27); filling a break or other empty time period (14/27), and the prize incentives offered by researchers (7/27).

The following paragraphs provide further detail on some common influences on motivation to engage with a display: curiosity and social embarrassment.

Curiosity

In their work on a design space for interacting with public displays, Múller et al. [MAMS10] identify curiosity and exploration as a key motivating factor. The significance of curiosity in motivating interaction has been observed in a number of prototypes.

Cao et al. [CMB08] reported that the majority of players of their game Flashlight Jigsaw were motivated by the novelty of the game. Furthermore, the nature of the game meant that the image on the screen was only revealed through gameplay meaning that curiosity about what the final image also engaged users. In their display system iCom, Agamanolis et al. deliberately designed for curiosity by displaying “cryptic subject lines... to motivate passers by to click and reveal their full text” [Aga03, p. 21-22]. More recently, Houben and Wiechel [HW13] have explored the use of a ‘curiosity object’ (a wooden box with a switch on one side) placed near the display as a mechanism for promoting engagement. During a one-day trials without the object, no interactions were observed; during a one-day trial with the curiosity object 41 interactions (with either the display or the object) were observed. 76% of those attracted by the object also interacted with the display.

Similarly, Goncalves et al. found curiosity to be strong motivator in their study of interactions with displays at the University of Oulu [GFH+13]. ~38% of interviewed participants report that curiosity was the main factor that lead them to interact with a display. However, their observations of ‘Unlockers’ (accounting for almost half of all interactions) demonstrate the limitations of curiosity as a motivator. Such interactions consisted solely of an individual touching a screen once in order “to see what happens” [GFH+13, p. 7] – the individuals then moved on past with no further engagement with the display.

Overcoming Social Embarrassment

Engagement with pervasive displays is a public, visible, action and embarrassment can be a significant barrier. The effect of social embarrassment on display interaction was first characterised in comments from trials of The Opinionizer (described in Section 2.2.6 in 2003 [BR03]. Those who interacted with the displays reported that they felt pressured to contribute socially acceptable comments and to avoid typing errors. Furthermore, over half of those interviewed who had not interacted with the display reported feeling embarrassment, for example one commented that they were “aware of other people watching, which made it kind of awkward” [BR03, p. 5].

Brignull and Rogers [BR03] suggest that allowing anonymous, remote, comments can provide a mechanism for reducing social awkwardness. They highlight the much higher rate of remote use of Greenberg and Rounding’s Notification

Collage (see Section 2.2.5.2 [GR01]) compared to co-present display use. Tang et al. promoted the use of personal mobile devices as a mechanism for supporting anonymous remote display interaction, thereby avoiding social embarrassment [TFB+08]. They found that whilst more users interacted with their MAGICBoard display through a nearby kiosk than through their mobile phone (ratio 5:2), phone users typically entered more content than kiosk users. However, both Tang et al. and Brignull and Rogers also note that the use of remote input methods also reduces or removes the honey-pot effect.

Cao et al. note that display users have varied attitudes to the presence of spectators [CMB08]. Their interviews with users of the Flashlight Jigsaw game on an interactive display provide further evidence that social embarrassment is a concern for users interacting with a public display. Approximately 40% of their study participants reported that they would distracted, nervous or intimidated in situations with many spectators. However, other participants reported that observation by others did not concern them, and still others reported that they enjoyed the presence of spectators. Some participants reported that they tried to perform better when observed, perhaps mitigating some of the potential for embarrassment.

Finally, Hardy et al. [HRD11] provide evidence that not all spectators have a negative effect on engagement. Their observations around a gesture-enable display suggested that social embarrassment was more likely to affect individuals who interacted with the display alone rather than those accompanied by a group. This is consistent with interview data from Cao et al. in which players of an interactive game reported that the presence of others made gameplay more acceptable (e.g. "I'm not too worried about doing it because other people are doing it too"[CMB08, p. 8] and said that they preferred playing the game with others rather than alone (25 out of 27 interviewees); usage logs also showed that players did play for longer when playing with others (average session length for one player was 6.5 minutes, two players 10.5 minutes, three players 12.2 minutes).

2.4.3 Interaction between Multiple Display Viewers

The presence of multiple people at a display elicits a range of behaviours. For example, we have already noted the honey-pot effect (Section 2.4) and examples of social learning (Section 2.4.2.3).

Numerous display systems have been designed to promote collaboration or social interaction between individuals (Sections 2.2.5.2 and 2.2.6), creating effects such as triangulation in which content on the display stimulates conversation between strangers [LMEA11, MEL11, MLA11]. In display-based games, collaboration may be seen alongside competitive play. For example, in their multi-player game Flashlight Jigsaw, Cao et al. [CMB08] noted a mixture of competitive and collaborative play with some individuals engaging in both styles at different points in time.

Turn taking around interactive public displays is common. For example, Michelis and Müller observed that when only one of their Magical Mirrors displays was active, one member of a group would interact whilst the remainder formed a circle around that user and waited for them to finish [MM11]. Similar behaviour was seen in earlier studies: during interactions with BlueBoard [RDS02] (see also Section 2.2.6) groups of potential users would wait around the display for their turn or all interact through a single user, a driver, who performed activities on behalf of the group. Peltonen et al. [PKS+08] report the use of 'terminal activities' as a mechanism for indicating to others when a turn is complete.

Not all social effects are positive. Hardy et al. [HRD11] noted that where an interacting user was part of a group at the display, the interacting user's attention tended to be focussed on the group rather than the display; players of a gesture-based asteroids game would stop steering their game ship in order to attend to a conversation within the group. Müller et al. reported that on occasions interacting users were so strongly engaged by the display that they became unaware of their neighbours resulting in unintentional collisions [MWB+12]. In the CityWall deployment, Peltonen et al. [PKS+08] report on cases in which users intentionally or unintentionally overlap interaction areas resulting in the withdrawal of one or both parties from the screen.

2.5 Personalisation and Appropriation

This thesis is focussed specifically on support for user personalisation and appropriation of public displays.

A small number of studies have explicitly explored the kinds of content viewers would like to see on a public display, often providing evidence in favour of display personalisation. As part of their work on Display Blindness (described in Section 2.4.2.1), Múller et al. [MWE$^+$09] interviewed ninety-one participants at eleven public displays in Münster, Germany. Each participant was asked about the content they would like to see on the display. Elicited content types included local information, local news, sports news and entertainment content. The authors note that desired content was very varied and that personalisation of the display may be a useful mechanism for serving these conflicted interests.

Memarovic et al. also interviewed participants to explore the kinds of content viewers would like to see on public displays [MLA11]. They interviewed seventeen students from the University of Lugano and found that participants were primarily interested in official Unviersity information and local events. They also noted that "students also wanted to be able to post their own content" [MLA11, p. 8] and highlight the role of user- contributed content in promoting community values.

In Section 2.2.7.3 we described research that used viewer profiles to optimise which advertisements appear at a display. Such systems use viewer interests to select an advertisement from a pool of items available to show at the display. Whilst this form of customisation clearly has commercial value, some research work has also explored more general approaches to display personalisation. In this section we develop our previous history of digital signage to provide further detail on prior research activities in this space.

2.5.1 Remote Desktop Support (1980s onwards)

From a traditional computing perspective, an early approach to user appropriation of display hardware can be seen in the form of desktop sharing protocols that allow any computer running appropriate client software to remotely access the graphical interface of another computer (i.e. to view the applications running on one machine on the display hardware of another). In this section we provide an overview of existing work in this space.

An early system for allowing network separation between an application and its user interface was the X Window System (X) [SG86]. X specifies the display as a server component and mandates that graphical applications (clients) connect to the display server. The server handles mouse and keyboard operations from input devices connected to the display and translates them into standard protocol messages (e.g. `KeyPress`) to be sent to the client application; equally the client application sends standard display operations (e.g. `XCreateWindow`) back to the server to be drawn onto the display. Extensions to X have been developed to reduce network traffic and improve performance; for example, NX [NoM14] provides compression and introduces a proxy to identify and cache common server responses to client events thereby reducing the number of network roundtrips (e.g. a menu that appears in response to a mouse click event).

Another system that allows remote access to graphical primitives (in contrast to alternative approaches that typically use pixel buffers) is Plan 9 [PPTT90]. The Plan 9 operating system is built on a set of principles that include the representation and access of all resources as a set of files accessed through a standard protocol. $8\frac{1}{2}$ [Pik91], the Plan 9 windowing system allows any application that could access the files (including applications on a remote server) to communicate with the windowing system using standard file system operations (open, close, read, write).

In Virtual Network Computing (VNC) [RSWH98], Microsoft Remote Desktop (RDP) [Mic14], Chrome Remote Desktop [Goo14], PCoIP[Ter14], and SPICE [Red09], applications run together with their graphical interface on a server. Pixel representations of the entire host's desktop or an individual application's interface are transferred over the network to the client; input devices at the client generate events (e.g. `KeyEvent`) that are forwarded to the server for processing. RDP [Mic14] provides support for a wider variety of input and output devices (e.g. printers, sound); support for sound is also included in SPICE [Red09]. A variety of approaches are taken to optimise frame buffer updates. For example, SPICE allows individual window content to be stored as 'surfaces' on the client in order to improve performance for windowtransformation operations, and the tight encoding extension for VNC [Kap01] allows compression of frame update data to improve performance in low bandwidth conditions. The limitation of PCoIP [Ter14] to LAN use only, has resulted in a protocol layered over UDP rather than TCP to allow large volumes of data to be transferred from server to client without the need for regular acknowledgements.

A slightly different approach was taken by the creators of Thinc [BKN05a] and ICA. Like VNC and similar systems, graphical processing takes place on the same computer as application execution. However, Thinc and ICA are both implemented as virtual graphics cards that interpret drawing operations and map them to protocol operations to be sent over the network. This implementation allows Thinc to provide optimisations for synchronised video streaming.

Most recently, the Net2Display protocol [VES09] proposed by VESA moves remote desktop protocol specification away from computing vendors to a specifications agency. An initial specification was released in 2009 and includes a minimal featureset designed to encourage interoperability.

The technologies described in this section are clearly suited to supporting viewer appropriation of displays but require viewers to have a clear idea about the forms of content they will want appropriate screens with, and to plan ahead to set up their applications for subsequent display on a public screen. This matches well with our 'active appropriation' usage model but is less well-suited to other models. Furthermore, none of the technologies were designed with large pervasive displays in mind and limitations include restriction of display resolution to that of the remote host (e.g. as in THINC) or to a fixed maximum (e.g. as in RDP), and poor performance for large display walls [SBA07]. Solutions for improving performance on large displays have been proposed (e.g. DVNC [SBA07]) but have typically focussed on providing support for desktop applications on display deployments rather than viewer appropriation. Nevertheless, remote desktop support has played a key role display appropriation research and many of the systems described in the next sections are built on one or more of these technologies.

2.5.2 Teleporting (1994)

Whilst not specifically intended as a pervasive display system, one of the earliest examples of display appropriation was Teleporting developed at the Olivetti Research Laboratory [BRH94, RBMH94]. The authors explored the idea of creating mobile applications by migrating a single user's application interfaces to a workstation display in order to enable the user to continue working at the new location. Teleporting used the X Window System [SG86] to support migration of user interfaces; each user of the system had their own X Window Server, a proxy server, to which all their applications connected. While the proxy server was not connected to a screen, keyboard or mouse, it allowed mobile access to applications by building a connection with another 'real' X server thereby moving the interfaces to a user's applications to a display close to the user themselves.

The system used Active Badges [WHaG92] as a means of detecting user location. By querying the Active Badge system's location database, a proxy server could determine which workstations were co-located with a particular user – the buttons on the badge could then be used to select between local workstations and to initiate the teleporting process.

Teleporting was in use for over three years at the Olivetti Research Laboratory and the ideas ultimately became part of Virtual Network Computing [ATT99, RSWH98, WRB$^+$97] in which graphical desktop sharing and control can be achieved in a platform independent manner over any network.

2.5.3 FLUMP (1996)

Perhaps the first intentional work on pervasive personalised displays was Finney et al.'s Flexible Ubiquitous Monitor project (FLUMP). Their departmental deployment (also described in Section 2.2.4.3) consisted of two stations with CRT displays at which personalised content could be triggered by the arrival of a user wearing an Active Badge [FWDF96, FDE12]. In addition to personalisation, limited interaction with the FLUMP stations was possible: users could page through their content using two control buttons on their Active Badge.

Each user selected their personalised content ahead of time by creating their own homepage using HTML extended with a small number of additional tags.

Processing of these tags by the FLUMP server's CGI program would result in the replacement of the tag (e.g. the tag `<FLUMP_TIME>` would be removed from the page and in its place would be an HTML text representation of the current date and time. Five such tags were initially supported, the `<FLUMP_TIME>` tag as previously described, plus:

- `<FLUMP_MAIL>`
  Replaced with the total number of new and unread email messages for the user. For new messages the sender and subject were also inserted.
- `<FLUMP_TIMETABLE>`
  Replaced with the user's next scheduled appointment.
- `<FLUMP_FYLDE>`
  Replaced with the status of the nearby 'Flyde Coffee Bar' (open or closed) plus the amount of time until it next opens/closes.
- `<FLUMP_MATT>`
  Replaced with the URL of that day's 'Matt' cartoon from The Telegraph newspaper's website.

Whilst some technical knowledge was required to create the pages, their placement in a Computer Science department meant this was not an insurmountable barrier to use. A total of twelve users were reported to have created such pages and used them to customise the screens [FWDF96].

2.5.4 GroupCast (2001)

Like FLUMP, GroupCast [MCL01, McC02, McC03] (see also Section 2.2.6) was a small-scale workplace deployment that identified passing users through infrared badges and personalised the display accordingly.

A single shared display was deployed in a common area of the Accenture Technology Labs. Approximately fifteen users created UniCast [MCL01] profiles consisting of a pool of items to be shown on that user's own office peripheral display (a UniCast screen [MCL01, McC03]). Available content included news headlines, financial data, weather, horoscopes, webcam output, reminders and web content. When a UniCast user passed the shared display, personalised content was selected from their UniCast profile and shown at the shared GroupCast display.

The GroupCast display was intended to trigger conversations between colleagues – as a set of users passed the display a content item would be selected from one of their UniCast content pools and displayed anonymously. No evaluation of the system was reported.

2.5.5 WebWall (2002)

Ferscha and Vogel's WebWall [FV02] (see also Section 2.2.7.2) used a set of 'service classes' to allow mobile users seamless access to web content over a public display. Each service class allowed access to different forms of content. For example, a Gallery class provided image and video display, and Banner and Auction classes allowed placement of commercial-style content. Whilst some service classes could be instantiated directly at the display (over SMS, WAP or other methods), most of the complex service classes were typically instantiated ahead of time through a web interface allowing easy customisation through the user's preferred input devices. Once complete, instantiation was saved and could be fetched onto the display wall as desired. Evaluation of the WebWall was not reported.

2.5.6 Dynamo (2003)

Dynamo [BIF+04, IBR+03] (previously described in Section 2.2.6) allowed users to share a variety of media over a public interactive surface composed of one or more displays arranged together either horizontally or vertically (allowing arrangements that act more as a tabletop surface in addition to traditional public display layouts) and a number of base interaction points (consisting of a wireless keyboard, mouse and USB slots); support was also provided for the use of laptops and PDAs as interaction points.

Once connected to a Dynamo surface, a user could carve off parts of the display for personal or collaborative use using a personalised telepointer (cursor) and access their media resources (i.e. those plugged into the USB slots at their interaction point) via a personal palette.

During a two week deployment of Dynamo in a UK high school, students used the screen share and show a variety of media (up to 500+ items per day with images being the most dominant media type). Both content contribution and control of what was visible on the display at any item were entirely controlled by the students allowing them to use the screen for a wide variety of purposes including asynchronous file sharing (either with a targeted set of individuals or to simply release the files to everyone who encountered the display), for performance-style sharing of personal or interesting media (videos, photo decks etc.) and group work/discussions.

2.5.7 Prospero, C4 & ProD (2007-2008)

Congleton et al. developed a series of frameworks for supporting personalised public displays (see also Section 2.2.5.2). The first such framework, Prospero [Con07], was deployed at the University of Michigan. The system was designed with a vision to "empower the public" and to render traditional broadcast display models obsolete by providing support for reconfigurable, publicly controllable, displays. The system used a combination of automatically generated and user-contributed profile information to determine which Display Modules (visual elements designed for the display, e.g. a Google map or Flickr slideshow) should be shown to a specified user. To trigger personalisation, users could log-in either by swiping their university ID card or by logging into the Prospero web backend on a nearby computer. In addition to allowing users to specify preferences about which modules should appear where on the display, many of the modules took input parameters to determine the content that appeared within the module (e.g. a search term, Flickr ID or location name). Prospero was deployed for approximately six weeks and the Flickr module was seen to be particularly popular. No formal evaluation was reported.

The Context, Content and Community Collage (referred to as C3C or C4) [MCH08] was designed to assemble data feeds associated with nearby users into a single shared collage. Users were detected and approximately located (far from or near to the display) using their Bluetooth device. Users completed an initial web registration phase, entering their Bluetooth MAC address and feed parameters (e.g. Flickr account name or search terms). Upon detecting a nearby user, the C4 system merged that user's feeds into the set of currently available items. Every seven seconds an item was taken from the currently available set and added to the display in a semi-random position. The resulting collage of items would contain items from multiple individuals, each one marked with metadata that detailed how the item had come to be added to the display. C4 was deployed over eight touch-screen displays in a Nokia Research laboratory for approximately ten months.

The Proactive Display (ProD) framework [CAN08] was designed to build on experiences from the earlier systems in order to provide a low-level model to support a range of proactive (personalised) displays. The framework consisted of a six-stage pipeline. The first three stages focussed on user presence: *Presence Detection* was implemented using Bluetooth (as in C4) and magnetic card readers (as in Prospero); *Presence Governance* determined which of the 'present' users should be prioritised at the display (e.g. based on proximity or arrival time); and *User Annotation* would pull in information about social relations (e.g. group membership, common interests). The next two stages related to content processing: *Content Nomination* involved the creation of a candidate pool based on each user's specified content feeds and a set of rules reflecting their preferences (e.g. to show specified content in a particular order or only to a subgroup of friends); *Collaborative Selection* provided a mechanisms for making decisions about how to combine each user's candidate pool to create a single set of content to be shown. Finally, the *Presentation* stage combined the content pools and prioritisation rules created by the earlier stages and laid out a set of content items onto the display. The authors used the ProD framework to implement a number of display applications including a rewrite of their earlier C4 system, a Cohesion Collage which displayed only items from the intersection of those from all current viewers of the display, and Nearby – a geocoded representation of user feeds displayed on a Google map.

2.5.8 Cocoa Buzz (2008)

Like C4, Cocoa Buzz [ET08] also provided personalised collage arrangements based on social feeds. The system could be shown on personal computers and public displays and generated a new collage every minute. Cocoa Buzz's collages were based on a set of channels (content sources) and each user could subscribe to channels to create their own line-up. Interviews with potential users were used to assemble a list of potential content sources (e.g. news, traffic and weather feeds, photo sharing websites) and user customisations (e.g. restricting photograph selection based on a particular search term). Unlike C4, Cocoa Buzz collages only contained content from one channel line-up, essentially restricting them to a single-user (the system was predominantly deployed on personal computers with just one shared public display). Preliminary observations of a pilot deployment suggested that users could successfully customise their collages but no formal evaluation was presented.

2.5.9 Bluemusic (2008)

Mahato et al. [MKHS08] developed their system Bluemusic (see also Section 2.2.7.2) to allow mobile users to customise features of their environment by using an encoded preference string as the Bluetooth name of their personal mobile device. A prototype implementation focussed on music selection in public environments, the device name encoded a user's preferred artists. A cross-cultural study with 135 participants (72% German, 28% Indian) included more varied scenarios including personalisation of advertisements and news items shown on a public display. The study showed significant differences in content preferences and privacy concerns between the two cultures. For example, German participants were typically less positive about the system as a whole and 76% considered it a threat to their privacy; only 29% of the Indian respondents were concerned by the privacy implications of the system and each a majority of participants were positive about each scenario (minimum 76% positive reponses for personalising background music in a shopping mall, maximum 92% for a personalised news scenario).

2.5.10 Instant Places (2008–)

The Instant Places deployments in Minho, Portugal have provided a range of mechanisms for supporting different degrees of personalisation. In 2008, Bluetooth device names were used to allow viewers to attach tags to public displays (e.g. 'I am here') [JOIH08] (see also Section 2.2.7.2). A user could also use the tag to create a link to their Flickr photostream. The authors deployed two visualisations of user interactions: one that provided a real-time representation of the currently detected device names (single tags or photo sets for each user in front of the display), and one that provided a more place-focussed view in which historical tags were able to persist. In both visualisations all users present at the display were represented simultaneously, with each user being allocated a portion of the screen. A trial of the system in a university campus bar ran for six weeks and was preceded by a four week Bluetooth scanning phase which was used as a baseline. The authors identified that the system prompted users to make their Bluetooth devices discoverable and to change their device names (presumably to interact with the system).

More recently, the Instant Places framework has been used to support self-expression on public displays through 'pins' and 'posters' [JPS+12, JPSM13]. In order to use pins and posters, an individual first creates an identity through the Instant Places website. Once registered a user can browse and select pins, and create and recommend posters. Pins are a method of encapsulating content related to a particular cause or brand that people might identify with (e.g. sports teams, music artists). Each pin is represented by a visual icon, tags, and a set of content sources associated with the pin (e.g. a Flickr photo collection or Youtube video channel). At the public display, the Pins application looks up currently checked-in users and then shows content from the resulting feeds. Content from multiple user's pins are shown together in a single slideshow. When no pins currently exist within a place the application cycles through a set of the most popular pins. Whilst pins provide a mechanism for allowing users to express personalisation preferences (like many of the systems discussed in this section), posters allow Instant Places users to create their own content items and submit them for display within specific locations. A poster is an image-based content item created through the Instant Places website. Created posters can be 'recommended' by a user to a display owner using a mobile application; the poster then awaits approval from the display owner before appearing at a display. A six-month deployment of pins and posters was conducted on ten urban displays. Twenty-one unique users checked in to display places, generating 193 checkins in total. Over twice as many users created posters (55) than added pins to their profile (27) and the authors noted that users often failed to associate the pins with the content appearing at the display. Posters were used for a variety of purposes, sport-related content was most popular followed by advertising and hobby-related content. Many of the posters were repurposed from other mediums rather than having been specifically designed for the display.

2.5.11 Cyber-Foraging (2008)

Cyber-foraging techniques can allow a mobile user to appropriate nearby hardware to complete tasks that are not easily achievable on personal devices. For example, cyber-foraging of displays might allow a user to transfer content from their mobile to a nearby screen in order to provide clearer or shared viewing. A number of systems have allow a user to carry data on their mobile device for later transfer to a shared computing terminal or public display (e.g. Elope [PBW05], Ghiani et al. ghiani12); display appropriation was typically not a focus of any of these works, but was provided as one of many supported forms of cyber-foraging.

Perhaps the most flexible support for such appropriation has been provided through virtualisation, often as part of a broader vision for cloudlet computing [SBCD09]. Wolbach et al.'s [WHCS08] prototype system, Kimberley, allowed users to create virtual machines (VMs) containing their personal applications and configurations. Their VM was decomposed into a base (containing the operating system and common applications) and a personal overlay (containing personal applications and configurations). By carrying the overlay on their personal mobile device, a user could quickly reinstansiate their VM on any compatible terminal or public display. During the period of use, the VM's display output would completely take-over the terminal or shared screen. Once the user had finished using the display, changes to the VM were added to the user's original overlay and sent back to their personal mobile device. Evaluation focussed on the performance aspects of the system although a scenario for display appropriation was provided.

2.6 Summary

Pervasive public displays have long been a rich vein of academic research. Early explorations began by using media links to connect viewers in two locations (e.g. Hole-In-Space [GR80], Virtual Kitchen [JVG+01]), and social interaction and awareness have continued to be a key theme for public display research. Within workplaces and social environments, shared displays have been shown to maintain awareness [BHS06, HM03], increase social interaction [BR03, CNDG03] and create feelings of community [CDF+05a].

More recently, longer-lived research deployments and commercial displays have moved into real-world environments (e.g. e-Campus [SFD06a], BBC Big Screens [BBC12], UBI-hotspots [OKL+10], INFOSCREEN [Str13]). In contrast to many of the research works before and since (e.g. FLUMP [FWDF96], BubbleBadge [FB99], CoCollage [FMP+09])), such deployments are typically used primarily as a broadcast medium with little scope for adaptation by users. Advertising is a key motivator and content source for modern commercial deployments. A number of research works have explored the use of profiles to optimise advertisement selection (e.g. [ABK+09]) and provided support for follow-on actions such as coupon acquisition (e.g. [MK09]).

The tension between unobtrusive, ambient displays and eye-catching, engaging displays has been a long-running research theme. Many early displays took the form of novel digital devices (e.g. office plants [AS03, BM98], water features [HHT99, IWB+98]) or ambient, aesthetically-pleasing panels (e.g. [HS03, RSH00]). More recently attempts have been made to provide engaging, interactive displays, often using the a viewer's personal mobile device as an input mechanism (e.g. [Bal05, BJB09, EBF+08, JOIH08]). Personal mobile devices also provide a useful mechanism for allowing viewers to take information away from a display (e.g. interesting news articles [MRS06], or classified adverts [AKB+11]), or to provide a clearer view of a display in a crowded or obstructed environment [CLMT12, LKS+11].

A number of techniques can be used to increase the levels of attention given to public displays. ‘Attract sequences’, short animations that demonstrate the potential of the screen are commonly used (e.g. [CMB08, PKS+08]), and images that travel towards the user may be particularly effective [FS03]. Sequences containing explicit calls to action (e.g. ‘Touch Me’) may also entice a user to interact with the screen [KOK+13]. Despite their popularity, observational studies suggest that the effectiveness of attract sequences and calls to action may be limited. Peltonen et al. [PKS+08] note that passers-by would stand close the display but still failed to observe it despite a moving attract sequence. Similarly, studies of the UBI-hotspot deployment found that introducing a call to action had no noticeable effect on interaction levels [OKK+12]. Reactive movement has been used in a set of recent trials – as a user approached the display the screen would transition to display new content [HRD11, OKK+12] or to actively reflect the viewer’s presence back to them (e.g. through mirrored video [MM11] or shadow images [MWB+12]). Whilst these techniques are often more effective at attracting attention, potential viewers may well have passed the display by the time they have processed what they have seen, requiring them to double-back before they can interact [MWB+12].

Support for personalisation has been a feature of some research systems, and has generally been achieved either through customisation parameters (e.g. providing search terms or identifiers for photo feeds [Con07, FV02, JOIH08]) or by allowing users to create custom content ahead of time [FWDF96, JPS+12]. Targeting content to viewer interests is also a common feature of pervasive advertising research [ABK+09, SKM09b], since more relevant adverts are more likely to result in sales. Systems to support cyber-foraging of displays (e.g. Kimberley [WHCS08]) provide a much more open method of personalisation in which a viewer can completely take over the display for their own purposes.

Despite the wealth of research, observational studies have provided significant evidence that displays in public spaces currently receive low levels of attention, a phenomenon referred to as ‘Display Blindness’ [MWE+09]. Passers-by typically expect the content shown on public-displays to be of little interest [MWE+09] resulting in few, short, glances at the content [HKB08, HRD11] and, in some cases, deliberate avoidance of the display [KOK+13]. Furthermore, the dominance of non-interactive screens means that users often fail to identify interactive displays, so-called ‘Interaction Blindness’ [OKK+12].

In this thesis we focus on supporting personalised, viewer-driven content in future open display networks. This chapter has described the background for this work by providing an overview of work in the area of digital pervasive display systems and applications, with particular attention given to those systems in which content is in some way influenced by the presence of a particular viewer. We have also highlighted the importance of engagement when deploying digital displays in public spaces, identifying interesting audience behaviour documented in existing research.

While some initial research has been conducted in the area of display personalisation, approaches have typically been tightly constrained with limited mechanisms for controlling content and small-scale infrastructure support. This thesis provides the first work in the space with an explicit focus on providing generalised support for display appropriation in large-scale deployments.

We continue in Chapter 3 by exploring the design space for personalised displays, suggesting possible usage scenarios and describing a number of probes we have conducted into the area of user appropriation/personalisation.

Chapter 3

Understanding Appropriation

3.1 Overview

The previous chapter described work which demonstrated that, despite their prevalence, public display systems do not always receive the intended viewer response. Many passers-by simply ignore the displays, display content is not attended to and has little or no impact on those to whom it was targeted. We suggest that one mechanism for increasing attention is to personalise content to the users themselves, allowing the content to reflect the interests of those passing by.

In this chapter, we provide an exploration of the design space for user appropriation/personalisation of pervasive public displays through a series of scenarios and probes. Such a design space includes both implicit and explicit personalisation; highly flexible and constrained user appropriation mechanisms; and consideration for a range of factors including usability, business models, and systems concerns. One key factor in the success of systems that support such user-driven content is trust and we provide an exploration of this area from the point-of-view of multiple stakeholders.

Our exposition of the design space uses a set of scenarios to demonstrate varied usage models for display appropriation, each followed by a description of its key features. We then present three probes that explore aspects of the design space through a mixture of measurement, user studies, real-world trials, and analysis of existing systems.

The chapter concludes with an analysis of the requirements elicited through the described scenarios and probes.

Work in this chapter has also been published in:

- Nigel Davies, Adrian Friday, Peter Newman, Sarah Rutlidge, and Oliver Storz. Using bluetooth device names to support interaction in smart environments. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, MobiSys '09, 2009 [DFN+09a].

- Sarah Clinch, Nigel Davies, Adrian Friday, and Christos Efstratiou. Reflections on the long-term use of an experimental public display system. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, Ubicomp '11, pages 133–142, 2011 [CDFE11a].

- Sarah Clinch, Jan Harkes, Adrian Friday, Nigel Davies, and Mahadev Satyanarayanan. How close is close enough? understanding the role of cloudlets in supporting display appropriation by mobile users. In *Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communication*, PerCom '12, pages 122–127, 2012 [CHF+12].

- Kiryong Ha, Padmanabhan Pillai, Grace Lewis, Soumya Simanta, Sarah Clinch, Nigel Davies, and Mahadev Satyanarayanan. The impact of mobile multimedia applications on data center consolidation. In *Proceedings of the IEEE International Conference on Cloud Engineering*, IC2E '13, 2013 [HPL+13].

The Bluetooth device name probe was also presented as a demonstration at HotMobile 2009 [DFN+09b] and the e-Channels analysis as a poster at Ubicomp 2011 [CDFE11b].

A summary of this author's contributions to the above publications and the work presented in this chapter is given in Table 3.1.

Probe	Author Contributions
Display Personalisation with Bluetooth Device Names	• User study to explore usability and use cases • Analysis of system and performance measures with regard to display appropriation
Flexible Display Appropriation Through Virtualisation	• System design and development • Design and execution of performance measures and user studies • Collaboration on data analysis
e-Channels: Exploring Display Ownership and Trust	• Collaboration on analysis of usage data for e-Channels • Collaboration on design and execution of viewer interviews • Analysis of viewer responses to e-Channel content

Table 3.1: A summary of this author's contributions to each of three design space probes.

3.2 Motivating Scenarios

In Section 1.2.3, we introduced the term *appropriation* to describe the process of placing *personalised* or *viewer-driven* content onto a public display. The following scenarios cover a range of usage models for appropriation and personalised content.

3.2.1 Ambient Information Discovery

Laney is a keen traveller who takes every opportunity to visit new places. She's recently come back from Hong Kong and is working all hours at her three part-time jobs whilst considering her next big trip: some friends have invited her to join them on a trip to Morocco but she also has an opportunity to go to Russia with an old contact. On a quick break for lunch she can't resist the urge to look in a nearby travel agent's window for inspiration. She glances at the last-minute package deals but nothing appeals to her. In the remainder of the window, a digital display shows flight costs, weather and other local information for a selection of worldwide locations. As she turns her attention to the display she realises that among the listed locations weather is supplied for Moscow (currently 1°C) and Rabat (currently 18°C) – she'd been researching hostels in these two locations only last night. As she walks away from the travel agent's she thinks some more about her options – a cheap trip to cold Russia, or a more expensive flight to warmer Morocco – it's certainly cold at home right now, perhaps a warmer climate would be nice.

This scenario provides a description of a usage model that we refer to as *walk-by appropriation*, in which a viewer passing a single display sees content that is relevant to them. This walk-by appropriation model can also be seen in the scenario in Section 1.2.2 when Jack's world clock application appears on a display near his workplace.

Walk-by appropriation is perhaps the most commonly accepted model for personalisation of displays – as demonstrated by FLUMP [FWDF96], Group-Cast [MCL01], research on personalised display advertising, C4 [MCH08] and the ProD framework [CAN08] (see Section 2.5). In addition to walk-by appropriation, we can identify two other appropriation usage models: *longitudial* and *active* appropriation – we will examine these later in this chapter.

In order to support walk-by appropriation, one or more methods of viewer detection must be available. In Section 2.2.7.2 we highlighted the significant role of personal mobile devices in previous research. Other viewer detection methods have been used (e.g. infrared Badges in FLUMP [FWDF96] and GroupCast [MCL01]), but mobile devices are currently the most prevalent mechanism. Such methods often focus around short-range communication technologies (e.g. Bluetooth). Later in this chapter we describe one probe conducted to explore the use of Bluetooth communication and personal mobile devices for supporting viewer detection.

Our scenario also demonstrates the transfer of current personal interests to the display or application in order to allow selection or generation of relevant content. Such transfer may occur *explicitly* (a viewer deliberately provides a list of interests or a one-off request) or *implicitly* (a viewer allows the display to build a profile of their interests, perhaps scraping content from other sources, or using sensing technologies such as cameras and eye tracking in order to detect which content has previously been of interest to the viewer). In this scenario, implicit transfer is inferred, with some connection back to Laney's recent web browsing history.

However the data is transferred, this scenario raises a number of privacy concerns. We identify two broad areas:

Disclosure through the construction of tracks and profiles This scenario demonstrates how integration of data associated with current viewers can allow a display to show personalised content. However, if we assume that the display in this scenarios is simply one of many such screens that can be customised, then specific data exchange risks arise. Exchanging information between displays allows content to be tailored for Laney wherever she goes, but also creates the potential for generating highly-detailed and revealing profiles that are shared across all possible displays.

Two specific risk types are introduced by these connected profiles:

Providing access to a greater set of information. For example, whereas Laney may previously have indicated her travel interests to display at a cultural event, and her music interests to a Jukebox display in a coffee shop, she could now potentially be revealing all her information to both displays.

Allowing new pieces of information to be inferred. The co-location of Laney with the physical environment in which a display is located is, in itself, extremely revealing. In an open display network, passing a specific series of displays in turn produces a trace that describes the way Laney is moving through the environment. A single trace could be used to predict an approximate destination, but the combination of multiple traces and timing information can be used to infer home and work locations, working hours etc.

Disclosure through the display of inappropriate or revealing content

The visual nature of pervasive displays, and their location in public environments, introduces a risk for disclosure through the display of a personal content item to those for whom it was not intended (e.g. passers-by, companions).

This disclosure can take a number of forms. For example, information shown might be inappropriate for any other individual to view (and so should only be shown when no-one else can view the display), or it may be that the content becomes inappropriate or revealing when shown in the presence of particular groups or in a specific environment.

When displaying communication exchanges (e.g. social network messages, chat, email, video windows), this disclosure risk not only affects the intended viewer of the message, but also the source (i.e. the message sender).

3.2.2 Cyber-foraging: Transient Display Use to Augment Mobile Hardware

Dr. Jones is at a restaurant with her family. She is contacted during dinner about a pathology slide that must be interpreted while surgery is in progress. Viewing the slide on her tiny smartphone screen would be useless. Fortunately, a large screen in the lobby (that usually displays advertising) is available for brief use by *customers. Walking up to the display, Dr. Jones views the slide at full resolution over the Internet. Using her smartphone for control, she is able to zoom, pan and rotate the slide just as if she were at a microscope in her lab. Privacy-sensitive clinical information about the patient is displayed only on the smartphone. Dr. Jones quickly interprets the slide, telephones the surgeon, and returns to dinner.*

This scenario is adapted from Wolbach et al. [WHCS08] and is an example of cyber-foraging for a display to overcome a lack of mobile screen real-estate (see also Section 2.2.7.2). In stark contrast to the previous scenario, here Dr Jones completely takes over the display (as opposed to simply tailoring an existing application). The scenario gives an illustration of our second usage model, that of *active appropriation*. In this model a viewer actively engages with a display, making a deliberate decision to appropriate a display to access a specific item of content.

Unlike the previous scenario, active appropriation involves the selection and display of a very specific set of content items. In this case, Dr Jones' needed to view a specific image, which was accessible for viewing over the Internet, and the restaurant display was open to accept that content. In a more constrained display system, selection of a specific content item (i.e. active appropriation) may take the form of navigating through a touch menu or other interface in order to select one application from a set of those available (e.g. as in Oulu's UBI-hotspots [OKK$^+$12]).

This decision about the degree to which to constrain viewer appropriation is one of considerable importance to display owners. Whilst a more flexible model has the potential to meet viewer needs more effectively, it also poses the greatest risk in terms of the display of offensive or inappropriate content. Furthermore, current commercial display deployments are typically advertising-driven and this model has the potential to keep advertising content off the screen for an extended period. On the plus side, this model is actively engaging a user with the display; something seen only infrequently for current deployments and a number of researchers have highlighted the difficulties of generating content for pervasive displays [SFD$^+$06b, TC09] – allowing users to generate their own content has the effect of reducing/distributing this burden.

Compared to walk-by appropriation, use of the display in this scenario illustrates a number of different qualities that distinguish active appropriation from other usage models:

1. Whilst interaction with a display may occur for content personalised under any of the usage models (and also for non-personalised content), content shown as a result of active appropriation is more likely to result in interaction than other items.

2. Active appropriation is likely to have a longer per-display engagement duration than other usage models.

3. During active appropriation, the engaged user has an expectation that their content will not be interrupted by other items. A switch away from the viewer's content has a greater chance of being highly disruptive during active appropriation when compared with other usage models.

This scenario again demonstrates the considerable role of mobile devices in future personalised display deployments. Once her slide is available on the screen, Dr Jones uses her smartphone as an interaction device "to zoom, pan and rotate" the screen content.

Finally, whilst other scenarios in this chapter demonstrate display appropriation for social or leisure activities, this scenario is focussed on use of the display to complete a work task, showing the range of possible future uses for personalised display systems.

3.2.3 Self-Expression: using longitudinal appropriation to express personality and creativity

Tom is a keen sports fan with a particular interest in football. He plays with friends in a local mini-league and supports Newcastle United, his local team when he was growing up. In preparation for a Sunday afternoon out at a local bar to watch the Tyne and Wear derby, Tom dresses in this season's Newcastle shirt and chooses from a set of his favourite football moments a collection of images to project onto nearby public displays.

As he walks to the nearby bar, Tom passes a few displays in shop windows and is pleased to see the club logo overlaid on some content items and a countdown until kickoff. He arrives at the bar and chats with friends as they wait for the game to start. Many of his friends support local team Aston Villa, whilst others support teams from their own hometowns including Liverpool and Blackburn Rovers; none support Sunderland. Tom is proud when some of his images appear on a display – as pictures from their different teams show at the display, he and his friends discuss how the season is going. Once the match is over, Tom and his friends agree to get food and then celebrate the result by visiting a number of their favourite bars and clubs. Over the course of the evening Tom and his friends pass many displays and Tom is pleased to see some Newcastle images appear in a range of locations.

Appearance and how others perceive us are important concerns for many. In this scenario we focus on the use of personalisation as a tool for self-expression and for promoting social interaction. Small-scale research trials have shown that displays can be an effective tool for raising awareness of shared interests and promoting social interaction (see Section 2.2.6); an open network of personalised displays offers the potential to offer this functionality on a wider scale. Self-expression is highly valued as a method of conveying personality and for channeling creativity. Projected displays have recently attracted attention as a medium for transient urban art (see Section 2.2.4.2); display personalisation can allow viewers to contribute to their environment, providing an outlet for conveying interests and temporarily personalising a space (cf. urban graffiti). Furthermore, the presence of many viewers can result in the generation of dynamic, collaborative art (as in Snow et al. [SJE11]).

This scenario also provides an example of our final usage model, *longitudinal appropriation*. In longitudinal appropriation, a user's preference to see personalised content is realised as an overall shift in the programming for a specific geographic area over an extended period of time. No guarantees are made that any one display will show content from a specific viewer at any particular time. By contrast, the aim of this usage model is to try and ensure that, within a given geographic region, the content that viewers see is more representatives of their interests than would be possible without personalisation. In this case, for example, the displays in this University town are likely to change significantly when the students are away for a time (i.e. during vacation).

Unlike the previous two scenarios, longitudinal appropriation is focussed on engagement with multiple displays for multiple viewers. Each engagement might be very short-lived and prone to interruption. Display interactions themselves are unpredictable and have a more serendipitous feeling than those created under other usage models.

Techniques for supporting personalised content from multiple users are numerous. In Section 1.2.2 we illustrated how, within a single application, the screen real-estate could be divided to show many users' clocks simultaneously. This 'space-multiplexing' technique divides a screen (or set of adjacent screens) into a number of portions, each allocated to a different user or application. Another common technique is 'time-multiplexing', in which, different users' content items are shown in sequence, each being allocated a portion of time in which the screen is dedicated to showing their content. Alt et al. [AMS12] describe a third method for presenting content from multiple stakeholders, that of 'integration'. Integration represents an alternative method of space-multiplexing where one content item is carefully embedded into another in order. For example, a slightly transparent user image may be overlaid on top of other content; or a number representing the current outdoor temperature may be placed inside an interactive game piece – this would allow one viewer to view their requested weather information whilst another continued to play the game they had selected. Each of these techniques also provides a straightforward mechanism for allowing personalised content to exist alongside traditional signage. Techniques may also be combined, for example by dividing the screen using space-multiplexing and then operating a time-multiplexing technique within each of those portions.

Finally, this scenario is the only one to highlight the role of personalised content as a method of generating content for the purpose of displaying to others. Common personalisation examples focus on using personalisation to meet a personal want or need (show me the time of the next bus, help me complete this work task), but here the focus is on generating content for others to see. Whilst this is perhaps a less-common scenario, it offers potential beyond self-expression. For example, a user might use personalisation to promote an event with which they are involved or to participate in some sort of social intervention (e.g. community promotion or protest).

3.2.4 Display Ownership in an Open, Personalisable, Display Network

Geoff owns a popular community cafe on a small street in the Northern Quarter of Manchester, UK. He's borrowed a small digital projector from a friend and is experimenting with using it to show interesting content behind the counter. When choosing content for his cafe, he starts by visiting the application store a nearby retailer had recommended. Browsing, he selects a set mixture of fashion, alternative music and creative applications he thinks his regulars might be interested in. To add to the store content, he also adds a few of his favourite websites to his content list. Over the next few weeks, many of Geoff's patrons comment on his projected display.

Ike, a cafe regular, watches the display as he waits for his cappuccino. He notices that some of his favourite design inspiration apps are appearing amongst the content. As he pays for the order, he asks Geoff if he would be interested in an application he's been working on. Ike isn't yet ready to release it into an application store but would like to try it in a real-world environment to see how it is received. Geoff agrees to trial the app and pass on any comments. Ike passes him the application details and takes his drink.

Like the previous scenario, this provides a demonstration of walk-by appropriation: content shown on the display is selected from a set of possible applications selected by Geoff, but is tailored to the cafe's current inhabitants so that during their visit each patron has an increased probability of seeing content of interest to them. Over the course of the day, as different groups of patrons enter the cafe, the overall theme of content shown on the display will change to reflect their interests.

Whilst previous scenarios were centred on the display viewer, this scenario focuses on two other key stakeholders: Geoff, a display owner; and Ike, a content producer. The scenario illustrates how a display owner may select content items from multiple sources; Geoff uses an application store (similar to those used on current mobile platforms) to select the bulk of his content and then augments this with his personal web bookmarks and an application contributed by one of his customers. Geoff's content includes simple media (music, images), websites, and rich content (e.g. applications). The scenario then demonstrates how personal interests of nearby patrons can influence which of the content items is shown at any particular time.

The scenario does not reveal Geoff's motivations for installing his projector. Müller et al.'s research on motivations for store owners to install digital signage [MWP10] found that many stores had received direction from a head office – for an independent store, no such direction would be given. However, Müller et al.'s research also found that the owners often felt that the displays would draw attention. Future personalised displays also have the potential to offer a service that might encourage a user to choose one cafe or store over another (similar to WiFi provision today); likewise the content shown on a display might alter the aesthetics of a space, acting in a similar way to pieces of art in current bars, restaurants, libraries and other social spaces.

From the content provider perspective, the introduction of viewer-customisable rich media types provides a new opportunity for developing a range of display applications. Research prototypes have included games [BGS+11, CMB08, FTLB08], community noticeboards [AKB+11, CNDG03, GMRS03] and file-sharing [BIF+04, SOC08, YTH+06].

3.3 Experimental Work

To explore specific issues and topics relating to user appropriation we constructed a series of probes. Each probe explored multiple aspects of the design space and together they cover a range of technical and human factors. As in the preceding set of scenarios, the probes also consider a range of different stakeholders.

3.3.1 Exploring Display Personalisation With Bluetooth Device Names

In this probe we explored the use of personal mobile devices as a platform for supporting display personalisation requests. As a result of their popularity and multi-functionality, personal devices are increasingly suggested as a platform for a variety of mobile computing applications. This study used on-board Bluetooth as a mechanism for detecting when a user had approached a display and as a platform for transferring personalisation requests. The Bluetooth device name technique was selected as a commonplace and free method of supporting requests.

A system for supporting display requests issued through Bluetooth device names was built by Peter Newman and Oliver Storz as part of the e-Campus system. In this probe a study of the system was conducted in order to evaluate usability and explore usage models.

To use the system, a user changed their Bluetooth device name to a string representing their personalisation request. Requests could trigger one of a set of pre-defined applications based on popular websites (e.g. Flickr, Youtube, Google) with a set of parameters to be based to those applications (typically search terms). This represents a relatively closed form of appropriation in which viewers could exhibit control over a display only within the constraints of the provided applications.

The Bluetooth system was centred around spontaneous appropriation. User registration was not required and content did not need to be prepared ahead of time. However, a TinyURL application gave some scope for selecting content beforehand if a user could anticipate wanting to appropriate a display with a very specific content item. With regard to our three usage models, this probe provides both active appropriation (a user approaches a display and requests their chosen content) and walk-by appropriation (a user sets their device name and then carries the device round for an extended period, viewing their content at different displays as they pass).

This probe was focussed on the viewer stakeholder, however the probe also unintentionally gave insights into other stakeholder's concerns. Our study predominantly aimed to explore the usability and usage models for personal mobile devices and Bluetooth device names as a platform for appropriating pervasive displays. We used task completion and survey methods for data gathering. In addition, we conducted measurements to gain some insights into the performance of our Bluetooth technique. Whilst the display owners had agreed in advance to use of their screens for the study, over time they became aware that the system was enabled but unsupervised during the evenings and weekends of the study period and expressed concern about the potential for misuse.

3.3.1.1 System Overview

Storz and Newman [DFN+09a] developed an e-Campus scheduler to support the submission of personalisation requests using Bluetooth device names. Personalisation requests could be submitted in the form:

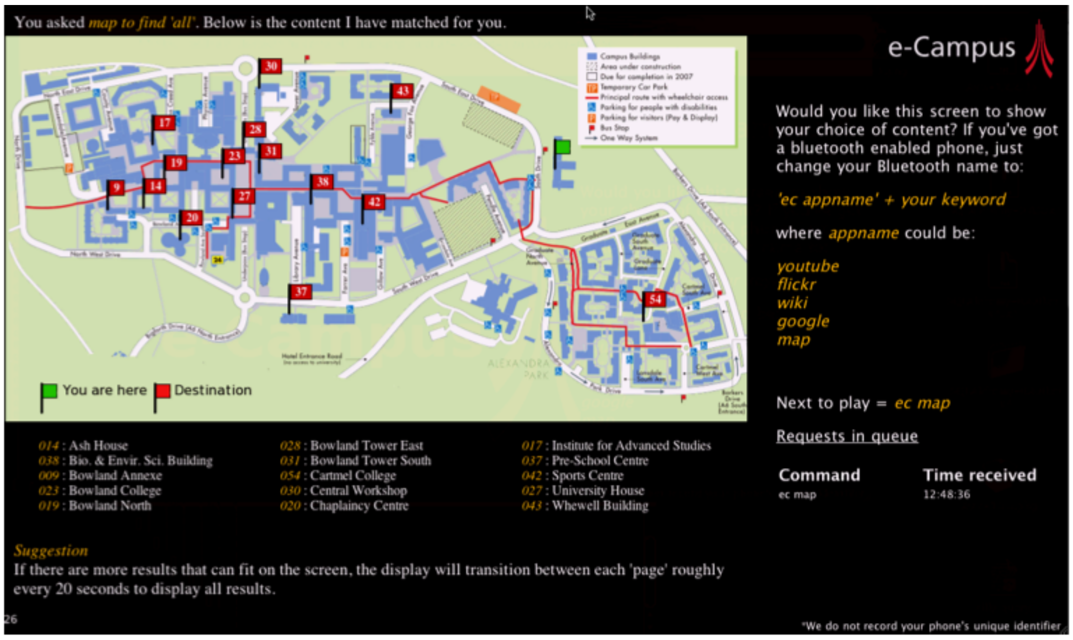
`ec ⟨application_name⟩ ⟨application_param⟩`

Valid applications included a jukebox (`juke`); a `map`; `tiny` (for selecting arbitrary web pages using the TinyURL lookup service; and `flickr`, `google` and `youtube` (for searching content on Flickr, Google, and YouTube respectively).

Upon receipt of a request (detected through Bluetooth scanning), the request was copied into a MySQL database and the e-Campus scheduler preempted existing content to launch a web renderer showing a PHP-generated web page. The PHP page read the personalisation request from the database and used the appropriate web service API (e.g. Flickr, Youtube, Google) to produce the custom content [Figure 3.1]. In the case of multiple requests being detected, a section of the PHP web page was given over to representing the queue of upcoming requests. A countdown timer shown in the corner of the web page indicated the remaining duration for the currently displayed item.

3.3.1.2 Experimental Method

To explore user experiences of the system we conducted a user study with twenty-four participants. The study was comprised of an initial survey about their current mobile phone use, followed by a session using the system, followed by a further set of survey questions about the system and their user experience. The study was conducted over a period of four days and the system remained avail-


You asked *map to find 'all'*. Below is the content I have matched for you.

e-Campus

🟩 You are here 🟥 Destination

014 : Ash House	*028* : Bowland Tower East	*017* : Institute for Advanced Studies
038 : Bio. & Envir. Sci. Building	*031* : Bowland Tower South	*037* : Pre-School Centre
009 : Bowland Annexe	*054* : Cartmel College	*042* : Sports Centre
023 : Bowland College	*030* : Central Workshop	*027* : University House
019 : Bowland North	*020* : Chaplaincy Centre	*043* : Whewell Building

Suggestion
If there are more results that can fit on the screen, the display will transition between each 'page' roughly every 20 seconds to display all results.

26

Would you like this screen to show your choice of content? If you've got a bluetooth enabled phone, just change your Bluetooth name to:

'ec appname' + your keyword

where *appname* could be:

youtube
flickr
wiki
google
map

Next to play = *ec map*

Requests in queue

Command	Time received
ec map	12:48:36

*We do not record your phone's unique identifier

Figure 3.1: Sample content item generated by the PHP application. Map content is embedded in a frameset that also includes a content timer (bottom-left), list of supported commands (top-right) and the currently queued requests (bottom-right).

able outside of the study hours on these days - a small number of requests were observed by users not participating in the study.

In addition to the user study, short trials were also conducted as part of a number of campus events. The trials received low levels of participation resulting in limited data.

Finally, Storz [DFN+09a] gathered a set of performance measurements that focussed on the duration of Bluetooth inquiry and name resolution using a Nokia N95 as a typical mobile device and a Mac Mini to represent our default display hardware. Sixty inquiry and resolution cycles were averaged.

3.3.1.3 Results

Our initial survey shows that all but one study participant had a mobile phone confirming that mobile devices are a good proxy for users. Seventeen had Bluetooth on their phone (74% of phone owners; 71% of participants), and thirteen were familiar with the process changing the device name (76% of Bluetooth phone owners; 54% of participants).

All participants successfully used the system to request content onto a public display, and all but one reported that the system was easy (13 responses, 57% of respondents) or moderate (9 responses, 39% of respondents) to use. Prior Bluetooth experience was not found to influence usability rating [Table 3.2].

	Easy	Moderate	Difficult	Impossible
Users with Blueooth	9	7	0	0
Users without Blueooth	4	2	1	0

Table 3.2: Participant responses to the question: "Overall, how easy did you find this service to use?" for the demonstrated Bluetooth device name system.[2]

When asked whether they would be likely to use the service, no clear consensus emerged. 35% of participants reported that they would be "likely" or "very likely" to use, 35% that they would be "unlikely" or "very unlikely" to use, and 30% gave a neutral response [Table 3.3]. Motivations for using the service included being able to look up information, the entertainment provided by using the system, the

[2]Table excludes one non-respondent.

novelty value of the system, and the potential for broadcasting information to others. Reasons for not using the service typically focussed on a lack of value from the service because the participant could not identify what they might do with it. Two participants commented that they would not use the Bluetooth technology on which it depended [Table 3.4].

	Very Likely	Likely	Possible	Unlikely	Very Unlikely
Users with Blueooth	3	3	5	5	0
Users without Blueooth	0	2	2	1	2

Table 3.3: Participant responses to the question: "How likely would you be to use this service?" for the demonstrated Bluetooth device name system.[3]

Positive	Entertainment	5
	Informative	4
	'Cool' factor	3
	Broadcasting to others	2
Negative	Lack of value from service	5
	Lack of interest in Bluetooth technology	2

Table 3.4: Summary of participant responses to the question: "Why/Why not [would you use a service like this]?" for the demonstrated Bluetooth device name system. Eighteen respondents, some responses included multiple motivations.

Overall our participants reported a preference for the Bluetooth method over a similar system using a mobile phone application (86% of respondents preferred Bluetooth, 14% no preference) or SMS (36% preferred SMS, 45% no preference, 17% preferred SMS) [Tables 3.5 to 3.7].

An exploration of potential applications that participants would use revealed a considerable preference for a map application over TinyURL (for arbitrary web content) and Facebook [Figure 3.2 and Tables 3.8 to 3.10].

None of our participants commented on the performance of the system although this did occasionally result in noticeably poor usability. Failed name resolutions were particularly costly at ~5 seconds, reducing performance in popular 'walk-through' spaces where devices quickly moved out of range.

[3]Table excludes one non-respondent.

Personalisation request method	Preference	Reason given	Number of responses
Mobile application (16 respondents, one with multiple motivations.)	Prefer Bluetooth	Application use is more effort	8
		Restricted by current device capabilities	2
		Doesn't match current download behaviour	2
		Non-specific opposition to downloads	2
		Application use is more complex	1
		Consumption of phone memory	1
		Application use is more expensive	1
SMS messaging (15 respondents)	Prefer SMS	More familiar with SMS than Bluetooth	4
		Bluetooth requires increased effort	1
	Neutral	I don't pay for my SMS messages	3
	Prefer Bluetooth	SMS messaging is more expensive	3
		SMS messaging is more effort	3
		Non-specific preference for Bluetooth	1

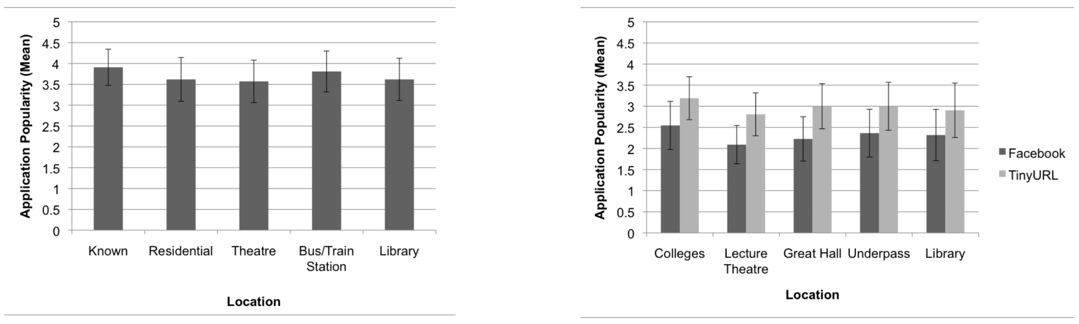
Table 3.5: Summary of participant responses to the questions: "Why/Why not [would you use the service through the specified alternative personalisation request method (mobile application or SMS)]?" for the demonstrated Bluetooth device name system.

	Device supports applications			Number of installed applications		
	Yes	No	Not known	None	1-3	4-6
No preference	0	1	2	2	1	0
Prefer Bluetooth (strongly)	11 (8)	4 (2)	4 (2)	12 (7)	4 (0)	3 (0)

Table 3.6: Preferred display personalisation request method (Bluetooth or phone Application) of participants whose phones did or did not support applications, and of who had installed varying numbers of additional applications onto their mobile phones. Note that none of our participants indicated that they would prefer to make display personalisation requests through a mobile phone application.

	SMS Payment Plan		
	None	Some	All
Prefer SMS (strongly)	2 (0)	2 (0)	0 (0)
No preference	4	2	3
Prefer Bluetooth (strongly)	2 (1)	2 (2)	3 (1)

Table 3.7: Preferred display personalisation request method (Bluetooth or SMS) of participants who paid for some, all or none of their SMS messages.



(a) Interactive map application

(b) TinyURL and Facebook applications

Figure 3.2: Popularity of personalised display applications at different locations

	Very likely	Likely	Possible	Unlikely	Very unlikely	No answer
A public area within the colleges	2	3	7	3	7	2
A public area next to a lecture theatre	1	0	7	6	8	2
The Nuffield Theatre/Great Hall	2	0	7	5	8	2
The Underpass (main bus stop)	1	4	6	2	9	2
The library	3	2	2	7	8	2

Table 3.8: Summary of participant responses to the question: “How likely would you be to use this service in the following locations?” for the Facebook scenario.

	Very likely	Likely	Possible	Unlikely	Very unlikely	No answer
A public area within the colleges	3	5	8	3	2	3
A public area next to a lecture theatre	2	3	8	5	3	3
The Nuffield Theatre/Great Hall	3	3	9	3	3	3
The Underpass (main bus stop)	3	4	8	2	4	3
The library	4	4	4	4	5	3

Table 3.9: Summary of participant responses to the question: “How likely would you be to use this service in the following locations?” for the TinyURL scenario.

	Very likely	Likely	Possible	Unlikely	Very unlikely	No answer
A public area within this University	7	8	6	0	1	2
A public area within a residential section of an area you did not know well	4	11	2	2	2	3
A cinema/theatre within an area you did not know well	3	12	2	2	2	3
A bus/train station within an area you did not know well	5	11	3	0	2	3
A public library within an area you did not know well	4	10	4	1	2	3

Table 3.10: Summary of participant responses to the question: “How likely would you be to use this service in the following locations?” for the map scenario.

A complete description of this probe including detailed method, results and analysis can be found in Davies et al. [DFN+09a].

3.3.2 Exploring Flexible Display Appropriation Through Virtualisation

In this probe we studied the use of virtualisation as a mechanism for providing highly flexible user appropriation experiences. We again used personal mobile devices as a method of initialising the requests (through use of a web browser to complete a simple form) and also as an interaction mechanism (using a mobile VNC/RDP application to send input to their applications).

Virtualisation has the potential to allow viewers complete control over the contents of a display by allowing users to customise their own virtual machine (VM). Viewers can configure their VM to run a particular set of applications in specific positions on the screen. However, this flexibility requires a significant amount of preparation - making a stark contrast to the previous probe which focussed on spontaneous appropriation. The use of virtualisation for display appropriation also provides benefits for the display owner in terms of providing security from malicious or poorly written viewer applications.

In this probe we explored the experience of the viewer stakeholder, with particular focus on the effect of application placement on interaction latency and the impact that this, in turn, has on usability. We also briefly explored start-up delays in order to establish how virtualisation might fit with our models of walk-by and active personalisation. We are not proposing that virtualisation as an effective means to achieve longitudinal appropriation.

3.3.2.1 System Overview

A virtual machine (VM) provides a safe environment for a user to execute their own choice of applications without impacting other users. Remote interaction and screen output can be achieved using VNC, RDP, SPICE and other similar protocols.

Virtual machines are typically composed of a set of files that describe disk and memory state. To instantiate a VM on a display, a viewer is required to create and configure a VM ahead of time. In order to execute their VM on our display appropriation platform, a viewer was also required to generate an overlay description that detailed how to recreate the VM using the disk and memory files. Descriptor files could then be transferred onto the viewer's mobile device to allow synthesis of the VM whilst on the move. Disk and memory image files could also be carried on the mobile device, or were required to be hosted on the web.

In order to allow interaction with the viewer's virtual machine at a display, the viewer would send the created descriptor file to Elijah, our system for display appropriation through virtual machine synthesis. The Elijah architecture is divided into multiple nodes: an Execution Manager, plus one or more Execution Hosts [Figure 3.3].

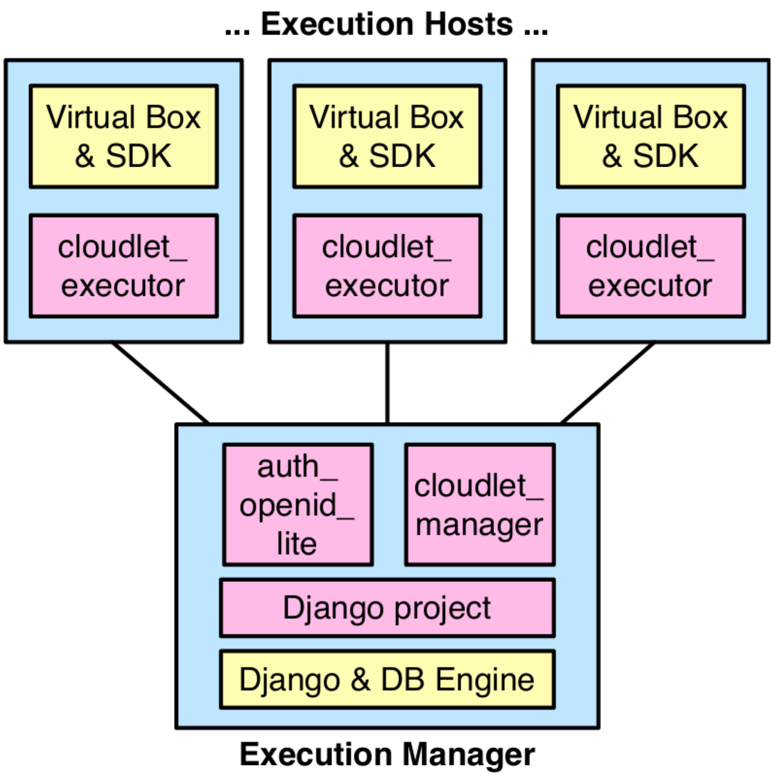


Figure 3.3: The Elijah architecture. Pink boxes represent portions of our implementation, yellow ones their dependancies.

The Execution Manager provides a web user interface (implemented using Python [Pyt13e] and Django [Dja13a]) that allows mobile users to submit VM requests in the form of an XML description that lists required disk images, state and configuration [Listing 3.1]. Once a request has been allocated to an Execution Host, VM synthesis is initiated and access details (i.e. VNC/RDP connection parameters) are sent to the user. A VNC/RDP connection can then be triggered at a display to show the output of the VM.

```xml
<?xml version="1.0" ?>
<virtualmachine>
  <state diskuuid="4edbe526-6d78-4ce9-885b-63dcce74de5c"
   diskpath="http://localhost/ubuntubase.vdi.lzma">
    <state diskuuid="3794437a-26a2-4702-8ec5-f7519b9b331c"
     diskpath="http://localhost/basewithabiword.vdi.lzma"
     memorypath="http://localhost/basewithabiword.sav.lzma" />
  </state>
  <memorysize>384</memorysize>
  <ostype>Ubuntu</ostype>
  <audioadapter controller="AC97" driver="Pulse"
   enabled="true" />
  <display VRAMSize="12" monitorCount="1"
   accelerate3D="false" />
  <USBController enabled="true" enabledEhci="true" />
</virtualmachine>
```

Listing 3.1: A Sample Elijah overlay descriptor.

An Execution Host is the specific physical computing node on which VMs will be executed. VM execution is achieved using VirtualBox [Sun09]. Each Host provides a set of VM slots, each representing the resources needed to execute one VM. Slots are allocated two ports, one for incoming launch requests and one for RDP/VNC data. Once a VM request is received at the Host, resources required to synthesise the VM (for example disk and memory images) are acquired, the VM is created and launched, and RDP/VNC connection information is sent back to the Execution Manager for delivery to the user.

3.3.2.2 Experimental Method

We conducted three experiments to explore the use of Elijah as a mechanism for supporting user appropriation of displays.

Two studies focussed on application placement and the process of interacting with an executing VM. We studied VM instances on our own hardware in the

UK, mainland Europe (EU) and Eastern US; and on the Amazon Elastic Compute Cloud (EC2) in Europe, Western and Eastern US and Asia [Figure 3.4][4]. We simplified an interaction scenario down to single key press event being sent from a mobile device, to the VMs using VNC, and benchmarked the time from the key press being transmitted, to the resulting screen update being received at a display [Figure 3.5]. Both the device generating the key presses and the display were located in Lancaster, UK. We benchmarked our VMs (using packet send/receipt times) in rotation during November and December 2010. A total of 11,277 measurements were recorded (~2,000 per Mac cloudlet and ~1,300 per EC2 instance).

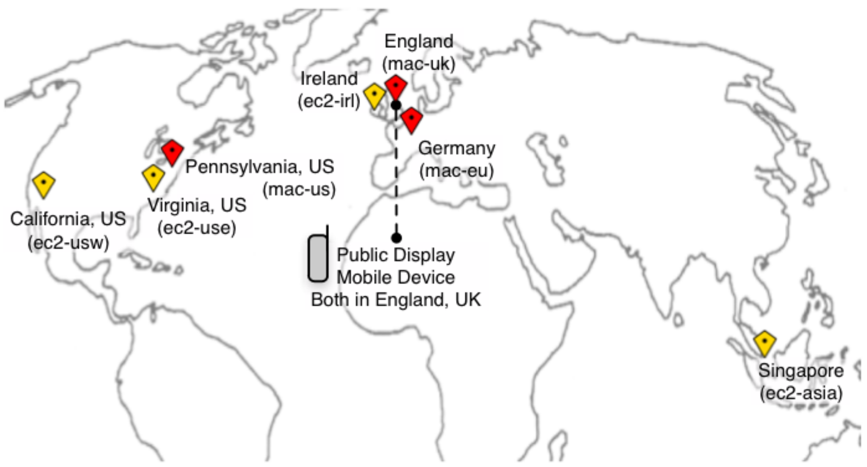


Figure 3.4: Geographic distribution of Mac cloudlets and EC2 nodes.

Our second study took a subset of these VMs (those on the Mac cloudlets) and used them as a platform for an interactive game ('whack-a-mole'), again using VNC for interaction [Figure 3.6]. We measured user performance (the number of moles hit and the time taken to hit them) and issued participants with a questionnaire that asked how about their user experience and the likelihood that they would play the game again if they encountered it on a public display. Twenty-nine participants completed the game and questionnaire.

[4]Our own VM instances were hosted on a Mac Mini (1.83GHz Intel Core Duo, 2GB RAM) running Mac OS X 10.4 and VirtualBox 3.0.12, and were allocated 1 CPU, 395MB RAM and 8GB disk. The cloud VM instances ran on the Amazon Elastic Compute Cloud (EC2) [Ama12] and were allocated 1 Compute Unit (1.7GB RAM, 160GB storage). Both sets of VMs were

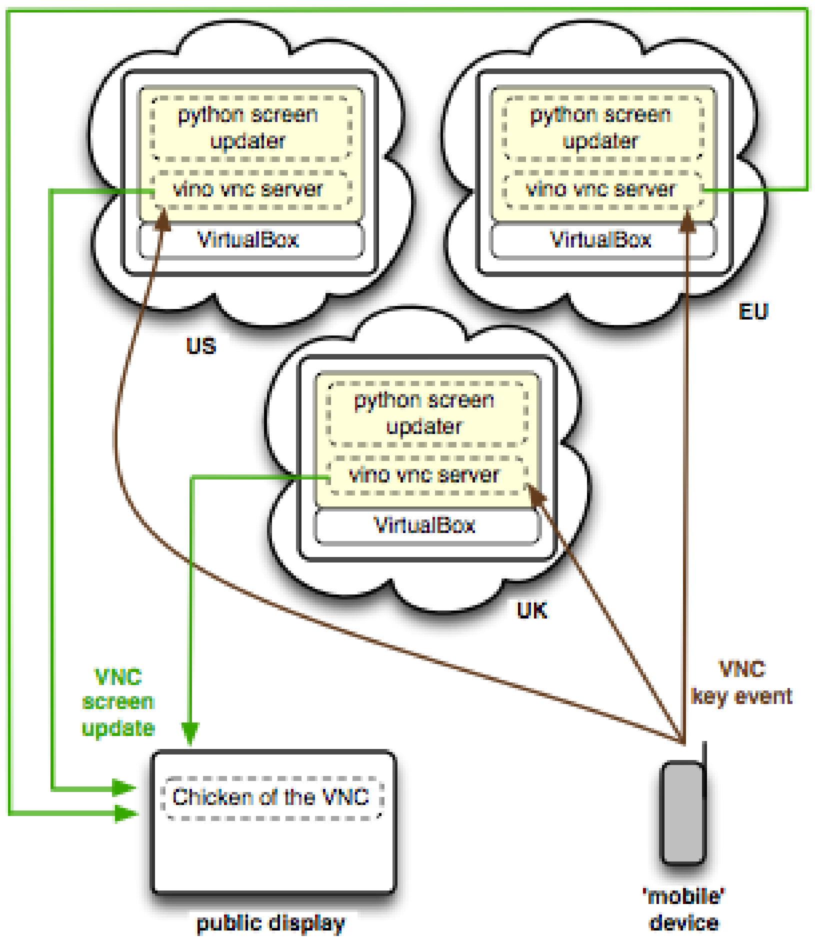


Figure 3.5: Benchmarking architecture.

Figure 3.6: The whack-a-mole game.

Finally, we completed a small set of stopwatch measurements that aimed to produce a ball-park figure for VM start-up times to establish which forms of appropriation virtualisation would be most suited to supporting. Our measurements were based on the synthesis of a VM running Ubuntu 9.04 and AbiWord.

3.3.2.3 Results

Tables 3.11 and 3.12 and Figures 3.7 and 3.8 present the distribution of the network and processing delays recorded during our benchmarking of the Mac cloudlets and Amazon EC2 instances. Median update times (network plus processing) are the most indicative of typical delays to be expected by application developers/users and are recorded as 60ms, 92ms, 171ms on the cloudlets (UK, EU, US respectively) and 90ms, 161ms, 227ms, 319ms on the EC2 instances (Ireland, East Coast US, West Coast US, Asia). Our Mac and EC2 update times are comparable for sites at similar network distances (EU Mac/EC2 Ireland, US Mac/EC2 East Coast). Using linear regression, we find that location has a significant effect on screen update time for Mac and EC2.

Significant outliers caused by TCP retransmissions are noted in all conditions [Figures 3.9 and 3.10] and skew the mean update times recorded in Tables 3.11

installed with Ubuntu 9.10, a VNC server, and Python 2.6.

Site	**Min**	**1Q**	**Median**	**Mean**	**3Q**	**IQR**	**Max**	**SD**	**SE**	**n**
Including TCP retransmissions										
mac-usa	122	157	171	345	188	31	23680	897.65	19.92	2030
mac-eu	40	77	92	257	108	31	15440	1046.41	22.79	2108
mac-uk	10	45	60	192	75	31	25940	1123.48	25.96	1873
Excluding TCP retransmissions										
mac-usa	122	155	168	186	183	27	1482	99.03	2.35	1770
mac-eu	40	74	88	93	102	27	1702	54.36	1.27	1828
mac-uk	10	43	57	59	71	28	527	27.99	0.69	1651
Cloud processing only										
mac-usa	6	38	52	69	66	28	1367	98.87	2.35	1770
mac-eu	10	42	56	60	69	27	706	38.88	0.91	1828
mac-uk	8	39	54	55	68	29	525	27.45	0.68	1651
Network latency (excluding TCP retransmissions)										
mac-usa	114	115	115	117	116	1	342	7.47	0.18	1770
mac-eu	10	31	31	34	32	1	1640	38.17	0.89	1828
mac-uk	1	2	2	4	3	1	120	6.26	0.15	1651

Table 3.11: Summary of Mac cloudlet update times in ms (min, max, mean, quartiles, standard deviation and standard error). Theoretical minima based on the speed of light to EU and USA cloudlets are 4ms and 39ms respectively.

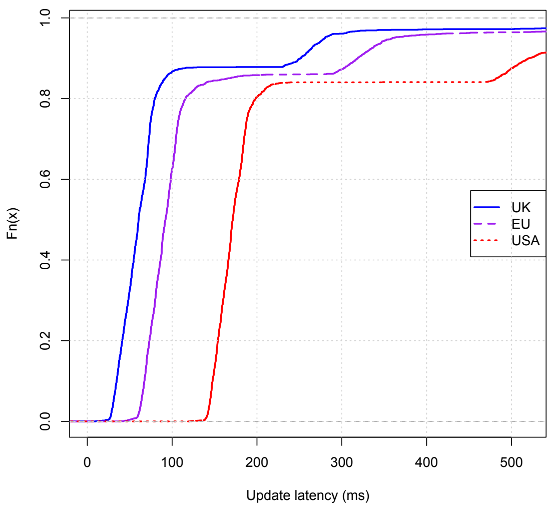


Figure 3.7: CDF of update response time (ms) by Mac cloudlet location. Only the bottom 95% (<533ms) is shown for clarity. The long tail (due to TCP retries) affects 13% of requests. The bottom 80% of the data is normally distributed (Shapiro-Wilk W=0.96, 0.98 and 0.98 respectively, p <0.001).

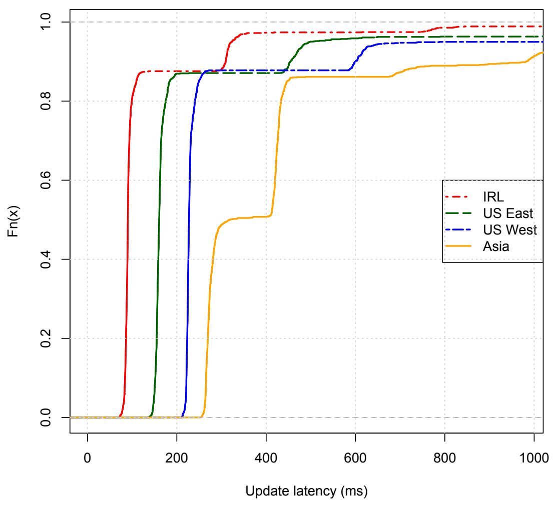


Figure 3.8: CDF of update response time (ms) by EC2 cloud location. Only the bottom 95% is shown for clarity. The long tail is again due to TCP retries and affects 12.8% of requests. An interesting feature is the step at 50% in the 'Asia' case.

Site	**Min**	**1Q**	**Median**	**Mean**	**3Q**	**IQR**	**Max**	**SD**	**SE**	**n**
Including TCP retransmissions										
ec2-asia	254	271	319	692	429	158	18160	1642.92	44.71	1350
ec2-usw	212	224	227	394	235	11	12750	917.49	25.26	1319
ec2-use	135	156	161	295	169	13	9956	738.24	20.21	1334
ec2-irl	72	87	90	187	96	8	14710	754.88	21.23	1264
Excluding TCP retransmissions										
ec2-asia	254	270	284	337	422	152	488	75.54	2.22	1162
ec2-usw	212	224	226	228	230	6	272	8.20	0.24	1158
ec2-use	135	155	159	161	164	8	227	9.30	0.27	1162
ec2-irl	72	87	90	91	93	6	137	7.05	0.21	1107
Cloud processing only										
ec2-asia	55	68	71	72	74	6	98	6.35	0.19	1162
ec2-usw	57	68	71	71	73	5	107	5.79	0.17	1158
ec2-use	51	67	70	70	73	6	105	5.72	0.17	1162
ec2-irl	51	66	69	68	71	5	84	4.39	0.13	1107
Network latency (excluding TCP retransmissions)										
ec2-asia	191	198	212	265	354	156	402	75.37	2.21	1162
ec2-usw	154	155	155	157	156	1	201	5.67	0.17	1158
ec2-use	82	88	89	90	91	2	152	7.43	0.22	1162
ec2-irl	19	20	20	22	22	2	71	5.67	0.17	1107

Table 3.12: Summary of EC2 cloud update times in ms (min, max, mean, quartiles, standard deviation and standard error). Theoretical minima based on the speed of light (in increasing order of Euclidian distance) are 2ms, 40ms, 52ms and 73ms.

and 3.12. Removing affected measurements provides a better model for our linear regression and reduces maximum update times (by 9.7–35.4 seconds) and standard deviations (by ~$\frac{1}{10}^{th}$ for Mac cloudlets and ~$\frac{1}{100}^{th}$ for EC2 cloud hosts). Single retransmissions are most prevalent (accounting for ~70% of all retransmissions), presumably due to the first wireless hop, but overall a greater number of multiple retransmissions are seen in those with greater network distance. With and without TCP retransmissions, the EC2 instance in Asia shows considerably greater variability than the others (SD ~2 $\times$ greater with retransmissions, ~10 $\times$ without; IQR ~10 $\times$ greater with, ~20 $\times$ without). Closer inspection shows that over 50% of the measures to this instance take significantly less time than the rest (median 272ms vs. 425ms). Isolating network and processing shows a 157ms reduction in network latency during December [Figure 3.11].

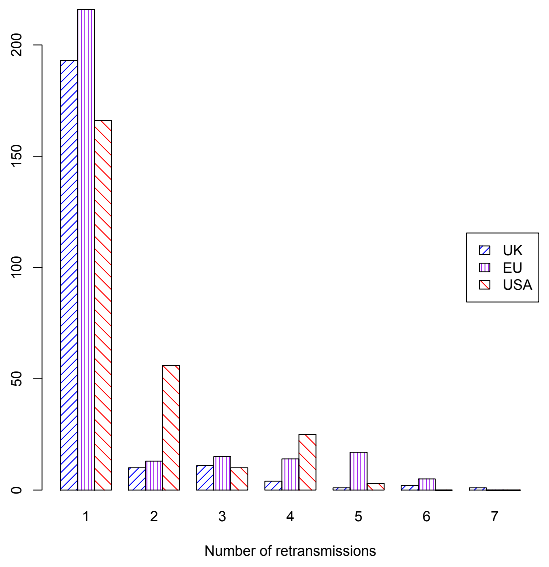


Figure 3.9: Comparison of frequency of retransmissions by Mac cloudlet location. Of the packets that make the first (wireless) hop, US has a greater proportion of 2nd and 4th level retransmissions.

User performance is described by the number of moles hit and the typical hit time in each condition. 87%, 85% and 71% of moles were hit in the UK, EU and US conditions respectively, with a significant difference found between the UK and US ($t = 6.29, p < 0.001$). Table 3.13 and Figure 3.12 summarise the hit times. Median hit time increases with the underlying latency (UK 179ms,

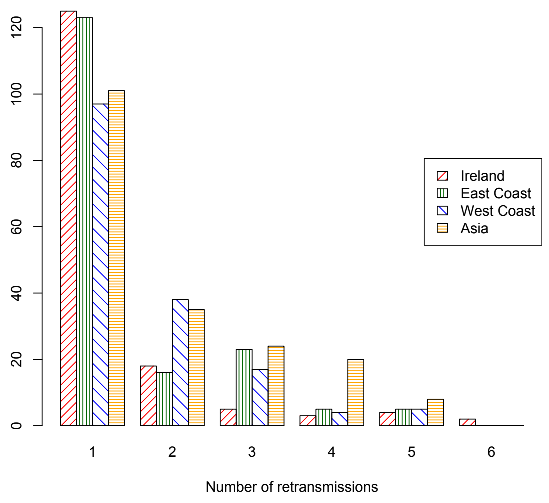
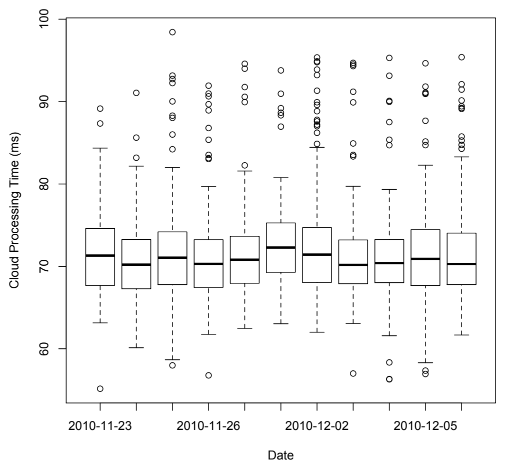
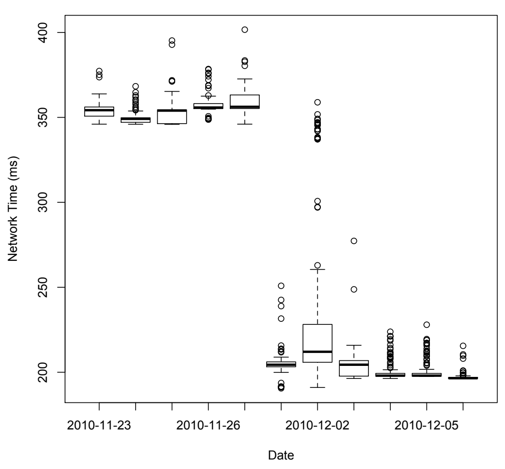


Figure 3.10: Comparison of frequency of retransmissions by EC2 cloud host location. The majority of losses are on the first (wireless) hop, beyond this paths with longer round trip times appear also to exhibit a higher degree of multiple failures.



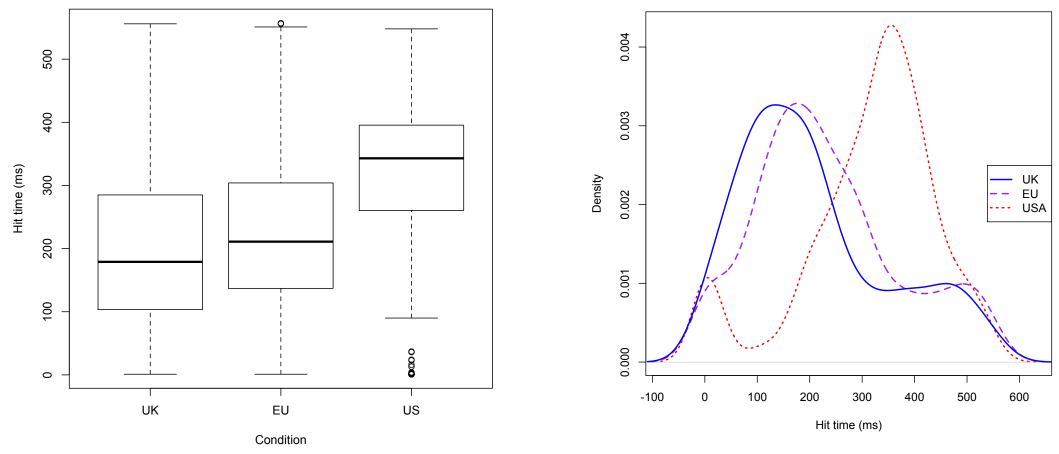
(a) Cloud processing time for the Amazon EC2 Instance, Asia.



(b) Network latency for the Amazon EC2 Instance, Asia.

Figure 3.11: Screen update time for the Amazon EC2 Instance, Asia (in milliseconds) by date (ascending, left to right) divided into (3.11a) cloud processing time and (3.11b) network latency. We observe that the EC2 is as responsive (processing) as other EC2 nodes. Before 1st December the median network latency is 355ms reducing to 198ms after this date.

EU 211ms and US 343ms) whilst interquartile range and standard deviation remain fairly consistent ($IQR = 132$–$179ms$, $SD = 126 - 142ms$). Using linear regression ($F_{2,1056} = 57.03$) we find that the EU takes 25.4ms longer than UK ($t = 2.55, p <= 0.01$) and US 108.5ms longer ($t = 10.35$, $p < 0.001$).



(a) Range of hit times experienced (b) Frequency distribution of hit times

Figure 3.12: Mole hit time by location. The US condition is visually distinct from the UK and EU.

Our user experience questionnaire showed that overall participants were positive about games hosted in the UK and EU, finding them sufficiently usable, responsive etc. to encourage further use (UK scores 43–52; EU 23.5–33[5]. The US game scored less well on all factors (scores -3–8), falling on the borderline of would/would not play again [Figure 3.13]. Ten users made comments that explicitly compared the behaviour of the games: three participants suggested that the EU and US conditions were both harder to control and/or slower to respond than the UK (e.g. "3 & 4 [Europe, US] noticeably lagger than 2 [UK]"); six that the EU and UK conditions were both easier to control and/or faster to respond than the US (e.g. "First games [US] seemed significantly slower to respond"); and

[5]All positive scores indicate that users would play the game (i.e. their reported usability/responsiveness/lack of frustration/sense of control over the game was greater than the reported point at which they would stop playing for that factor); negative scores indicated the converse. Theoretically the maximum score (i.e. the maximum difference between reported value and stop point) is ±120 although this would require the points to be marked at opposite ends of the line.

Site	**Min**	**1Q**	**Median**	**Mean**	**3Q**	**IQR**	**Max**	**SD**	**SE**	**n**	**Misses**	**Misses (%)**
mac-usa	1	263	343	316	395	132	548	126.35	6.06	435	126	29.0%
mac-eu	1	137	211	233	304	167	557	138.81	6.66	435	64	14.7%
mac-uk	1	104	179	207	284	180	556	142.48	6.83	435	55	12.6%

Table 3.13: Mole hit time by location (min, max, mean, quartiles, inter-quartile range, standard deviation and standard error).

one that the UK was easier to control and/or faster to respond than the EU which in turn was faster/easier than the US. Informal comments made during the study indicated that some participants attributed poor US performance to personal factors ("I was much slower to react"), scoring the games equally on the questionnaire.

Whilst synthesis time was not the focus of this probe, rough timings put VM startup at approximately twenty seconds without optimisation, suggesting that virtualisation would be a valid approach for supporting active display appropriation. Virtualisation may also be able to support walk-by appropriation, but this would be highly dependant on the range of the detection method.

A complete description of this probe including detailed method, results and analysis is given in Clinch et al. [CHF+12].

3.3.3 e-Channels: Exploring Display Ownership and Trust

In this probe, we again studied usage of an existing system, the e-Channels system described in Section 1.4.4. e-Channels is used as a mechanism for distributing content within e-Campus (described in Section 1.4) and it allows each display owner to select content from multiple University entities. The system therefore provides a useful reference point for future display networks with multiple content sources.

Developing open display networks in which content is sourced from multiple places presents both technical and social challenges. For example, cooperation from those that own and manage displays and spaces will be a key factor in the success of such a system. From a display viewer perspective, content interpretation in an open network poses a greater challenge than current networks as the source cannot easily be identified. When a source cannot be identified it may be difficult for the viewer to know how much trust to place in the message conveyed. In this probe, we explored digital signage use, display ownership and content interpretation through study of e-Channels.

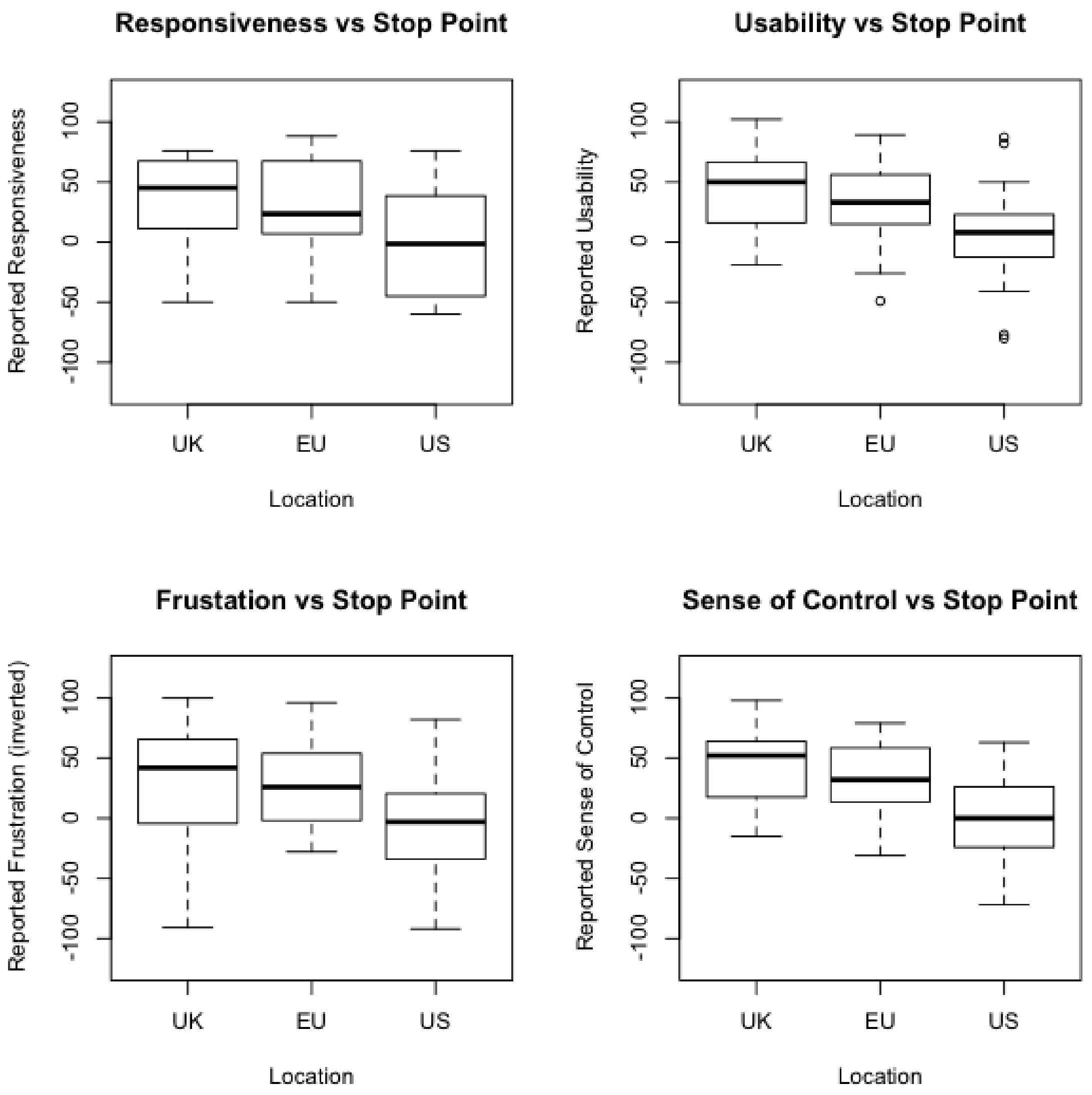


Figure 3.13: User perception: positive values indicate users would play the game (reported value greater than stop point); negative values the converse.

3.3.3.1 Experimental Method

Two studies were conducted. In the first study we used data from e-Channels to explore the system usage practices of display owners and content providers; in the second study, viewers were asked about items shown by the system in order to explore content interpretation.

Our first study used logs, file system traces, database queries and examination of digital content items in order to provide a descriptive data set that represented usage practices for display owners and content providers from the Channel System's first use in May 2008 until April 2011. This data provided a general overview of how both display owners and content providers used the system. A more trust-focussed analysis was then conducted on three snapshots of database records that described non-owner subscriptions to public channels. These snapshots were taken on 30th April 2012, 3rd May 2012 and 12thAugust 2013. These three points represent regular term time activity, an arts event held during term time, and regular vacation activity respectively.

To understand viewer perceptions of content within e-Channels, we interviewed thirteen participants (mostly students) recruited using opportunity sampling in three locations at which University digital signage was well-established. The study was conducted in April 2012 and each viewer was asked about a small number of content items taken from the set of all content items that were currently available at that display. Viewers were asked to identify the creator and owner of the content, to identify whether they trusted the content and considered the information conveyed by the content to be true, and whether they felt the content was acceptable for the space in which it was shown. A total of forty-one content responses were received (an average of three per viewer).

3.3.3.2 Results

Between 14th May 2008 and 11th April 2011, a total of eighty-one individual users in thirty-three user groups had contributed 3,700 content items (1,796 unique items) to 102 channels. Many of the user groups had longstanding experience with the system, with over half having used the system for in excess of a year [Figure 3.14].

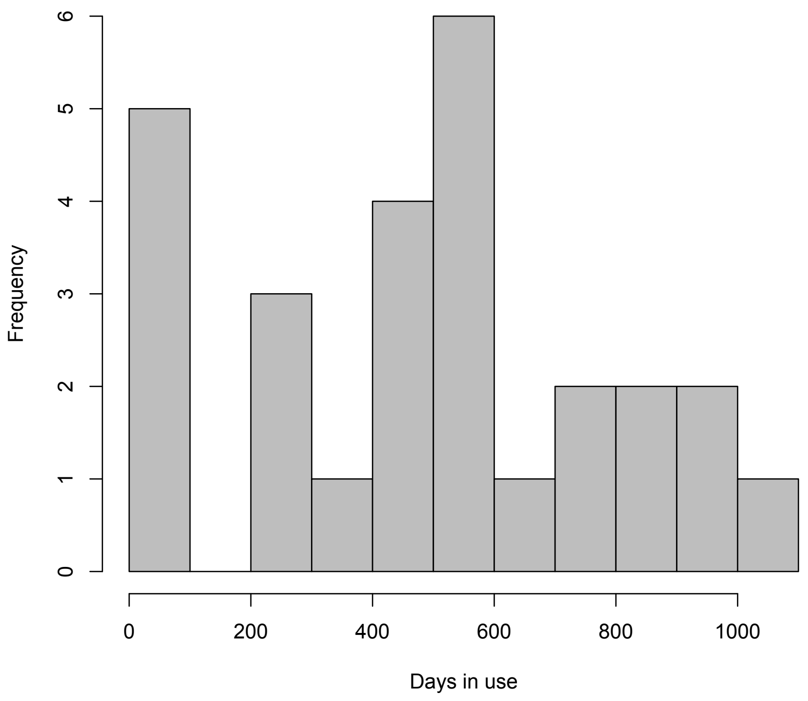


Figure 3.14: Breakdown of length of use of the channels system (from date of first piece of content to most recent) for active user groups in the system on 11^{th} April 2011. More than half the twenty-seven active groups have used the system for a year or more, eight or so over two years and new groups continue to be created.

Between 14th May 2008 and 11th April 2011 there were almost exactly equal proportions of shared and private channels created, with each content provider typically maintaining a small number (1-2) of active channels. Overall, the amount of content added each day was small (10s of files), with most files remaining in the system for either 7–10 days or 14-21 weeks [Figure 3.15].

Images were by far the most dominant media type [Figures 3.15a and 3.16], representing ~83% of unique files. Sampling one hundred image files, we found that most were specific to Lancaster University (95%), served a single clear purpose (91%) and appeared to have been explicitly designed for the displays (80%). Similar patterns were seen in the 128 videos and six media streams analysed. By contrast, from the sample of fifteen web pages, only 20% laid content out in a way that suited the resolution of the display. Finally, a high proportion (76%) of the sampled images demonstrated a noticeable style or identity and 55% contained a textual or iconic reference explicitly telling the viewer who the identity belonged to.

Looking at our three snapshots, we see clear evidence of display owners choosing to subscribe to other providers' channels, resulting in unknown items showing at their display [Tables 3.14 to 3.16]. On our three snapshot dates, the majority of displays sourced some or all of their content other providers channels (85%, 92% and 77% in April 2012, May 2012 and August 2013 respectively); a surprisingly high proportion (56% April 2012; 50% May 2012; 23% August 2013) sourced their content entirely from these channels.

Most non-owner subscriptions seen within e-Channels pull content from a small subset of available content providers [Table 3.14]. A very small set of sources remain popular across our three snapshots: the University Press Office, Student Experience office and Students' Union at 26–34%, 19–24% and 9–14% of non-owner subscriptions each respectively. Another small set demonstrate temporal properties, spiking in one or other of the snapshots. For example, during the arts event in April 2012, subscriptions to arts content climbs to 37% of non-owner subscriptions (with zero non-owner subscriptions before and after). Similarly, during the summer vacation 2013, subscriptions to Careers content rise to 17% of non-owner subscriptions from ~1% in previous snapshots, and subscriptions to Visit Day content increase to 18% from <3%.

	Term Time, Spring 2012		Arts Event, Spring 2012		Vacation, Summer 2013	
Content Source	Channels	Subscriptions	Channels	Subscriptions	Channels	Subscriptions
Public Arts	0	0 -	17	38 (37.25%)	0	0 -
Press Office	2	27 (34.18%)	2	27 (26.47%)	2	22 (26.83%)
Student Experience	1	19 (24.05%)	1	19 (18.63%)	1	19 (23.17%)
Students' Union	4	11 (13.92%)	2	9 (8.82%)	2	7 (8.54%)
Chaplaincy Centre	5	5 (6.33%)	0	0 -	0	0 -
Careers	1	1 (1.27%)	1	1 (0.98%)	1	14 (17.07%)
UKRSO (Visit Days)	2	2 (2.53%)	1	1 (0.98%)	1	15 (18.29%)
Other ([8, 5, 3] Owners at <[4, 2, 2.5]% per Channel)	10	14 (17.72%)	5	7 (6.86%)	3	5 (6.10%)
Total	25	79	30	102	10	82

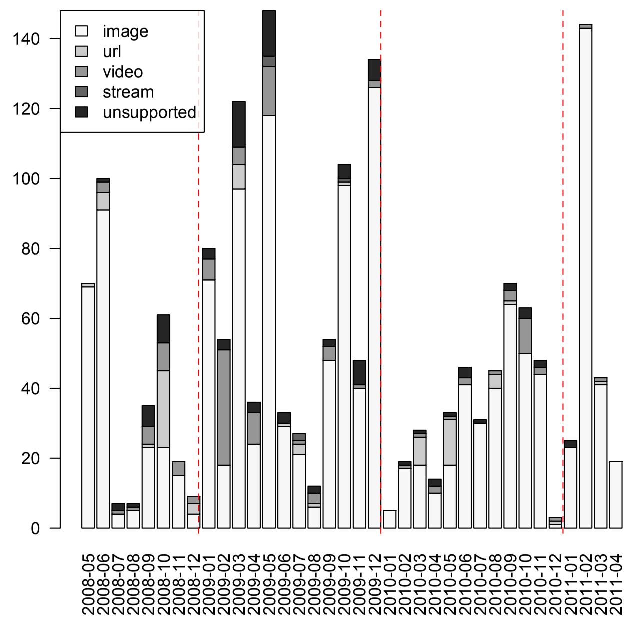
Table 3.14: Non-Owner Subscriptions to Public Channels (by Channel Owner).

	Subscriptions to channels with unknown content		Subscriptions to channels with known content				
	Content shared by others		Own content shared with others		Own content not shared with others		**Total**
Term Time, Spring 2012	63	(76.83%)	8	(9.76%)	12	(14.63%)	82
Arts Event, Spring 2012	102	(82.26%)	10	(8.06%)	12	(9.68%)	124
Vacation, Summer 2013	82	(69.49%)	14	(11.86%)	22	(18.64%)	118

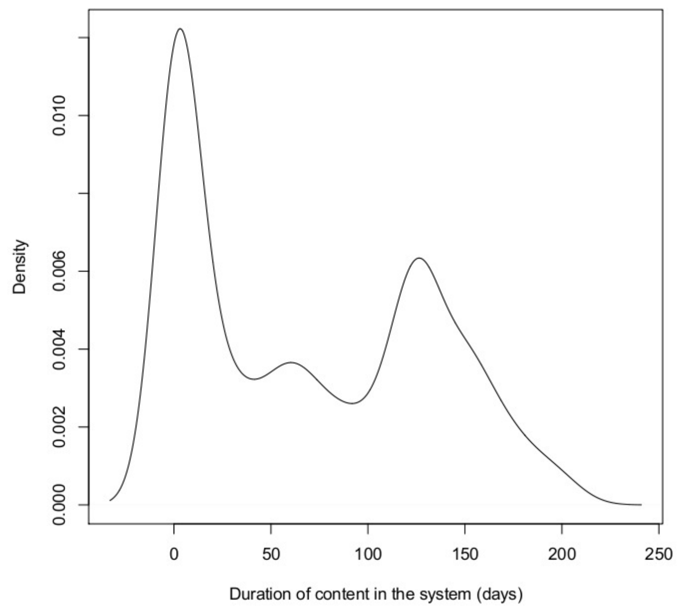
Table 3.15: Snapshot of e-Channels subscriptions in Spring 2012 and Summer 2013.

	Displays subscribed to known content only		Displays subscribed to a mixture of known and unknown content		Displays subscribed to unknown content only		**Total**
Term Time, Spring 2012	5	(15.15%)	10	(30.30%)	18	(54.54%)	33
Arts Event, Spring 2012	1	(2.94%)	16	(47.06%)	17	(50.00%)	34
Vacation, Summer 2013	8	(22.86%)	19	(54.29%)	8	(22.86%)	35

Table 3.16: Non-Owner Subscriptions to Public Channels (by Channel Owner).

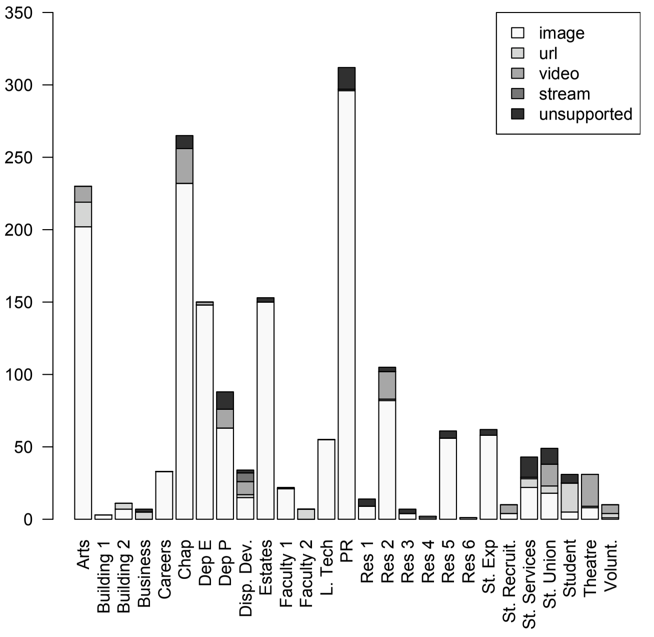


(a) Profile of files and file types added each month.

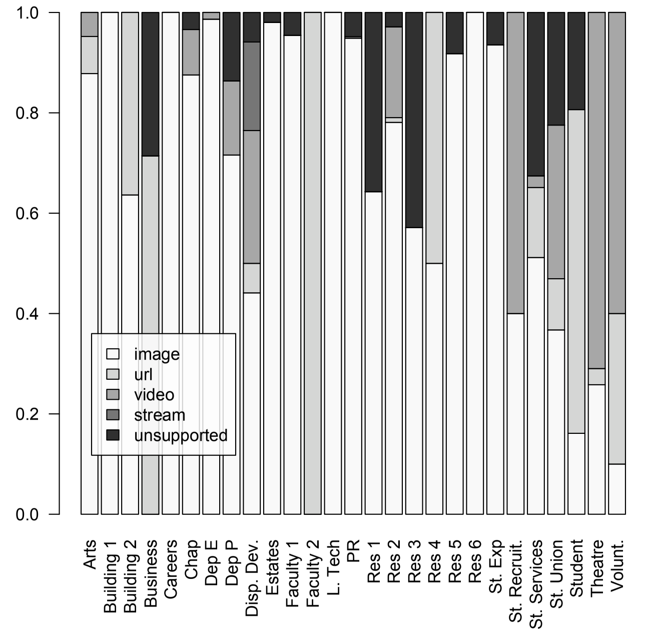


(b) Lifetime of content items.

Figure 3.15: Volume and lifetime of content within e-Channels between 14th May 2008 and 11th April 2011.



(a) All content items (count)



(b) All content items (ratio)

Figure 3.16: Content items added to e-Channels between 14th May 2008 and 11th April 2011 by provider group.

Our interviews with display viewers showed that 62% could correctly identify the owner of a display. Accuracy of content item attribution varied [Table 3.17]. Items produced by the display owner typically received very accurate responses (83–100% accuracy) but viewers were also able to accurately identify content from the Students Union (83% accuracy) and, to a lesser degree, the University Press Office (69% accuracy). However, one content provider's content items consistently yielded poor attribution rates, only 17% of content items from the University's Student Experience were correctly attributed to them. Perhaps unsurprisingly, content containing a textual or iconic representation of the source generally received more accurate attributions. Likewise, branding was often associated with accurate source attribution, except in the case of Student Experience content where viewers failed to recognise the brand.

Content Source	**Produced by Display Owner?**	**Includes Source?**	**Branded**	**Accuracy of Source Attribution**
Pendle College	Yes	No	No	100%
Students Union	No	No	Yes	83%
Engineering Dept.	Yes	Yes	No	83%
Press Office	No	Yes	Yes	69%
Student Experience	No	No	Yes	17%

Table 3.17: Factors influencing accuracy of source attribution by owner (most rated content items).

Trust ratings varied across the content sources. University press office content was considered particularly trustworthy (2 items; 15 ratings; 84% indicating strong levels of trust). Our most unfamiliar content source also received poorer trust ratings (1 item; 6 ratings; 33% strong trust, 33% unsure, 33% not trustworthy). Some of our viewer comments indicated factors that influenced their decision making with regards to trust. For example, the lack of an obvious source for the Student Experience content was a factor for one participant who noted "Where is the source?" alongside their indication that they did not trust the content. Overall 66% of content ratings indicated strong levels of trust, 5% trust with a some uncertainty, 20% uncertain, 2% who felt there was some truth in the content but it had been exaggerated and 7% who felt the content was not trustworthy.

A complete description of this probe including detailed method, results and analysis is given in Clinch et al. [CDFE11a].

3.4 Analysis and Requirements

3.4.1 Analysis

A number of themes reoccur throughout our scenarios and probes. As an overall motivation, we find that personalisation could be used for a wide-range of reasons. Our scenarios suggest ways in which personalisation could be used for exploring personal interests, completing an unanticipated but important task, and for self-expression. Usage suggestions from the Bluetooth probe included looking up information, entertainment, and broadcasting information to others. During this probe however, some participants struggled to identify how they might use personalisation—perhaps due to unfamiliarity. When asked about specific applications though, our viewers indicted considerable preference for a map application (utility content) over Facebook (social networking).

Returning to our viewer stakeholder, we identified three distinct usage models for appropriation in open pervasive display networks. The first two models, *Walk-by appropriation* and *active appropriation*, focus on the lone user. In walk-by appropriation, a viewer passing a single display sees content that is relevant to them. In active appropriation a viewer actively engages with a display, making a deliberate decision to appropriate a display to access a specific item of content (and potentially to interact with it). Our third model, *longitudinal appropriation* realises one or more space users' personalisation preferences through an overall shift in the programming for a specific geographic area over an extended period of time.

Our usage models also demonstrate different characteristics in terms of duration of use, and the degree to which interruption is acceptable. Based on our scenarios and probes, we suggest that active appropriation is the most demanding—viewers are typically likely to engage with their content for longer and in a focussed way that would be negatively impacted if their content were interrupted. For example, Dr Jones would have struggled to interact with and analyse her slide if it were periodically removed from the screen to show other content, or if she had been allocated an uninterrupted but short time period during which she could use the display. Likewise, our 'whack-a-mole' game players would undoubtedly have struggled in such circumstances. By contrast, longitudinal appropriation (and to a lesser degree, walk-by appropriation) are likely to result in short-term display use that can be interrupted with few negative consequences for the viewer.

Display applications will undoubtedly take many forms, and given current trends in cloud computing, may be hosted at a wide range of locations. Our study of user interaction with cloud- and cloudlet-based display applications as part of the second probe suggests that application location will impact user experience but that applications do not necessarily have to be co-located with the display to provide acceptable usability.

In order to support any personalised content, we note that some infrastructure must exist to support the detection of viewers and, for active appropriation particularly, the communication of requests. In our scenarios, this method was not described, but we noted the prevalence of personal mobile devices as a tool for display interaction in prior research. In line with this prior research, both our Bluetooth and virtualisation probes made use of mobile phones. In both probes the phone was used to communicate the personalisation request. In the virtualisation probe the phone was also used to support ongoing interaction with the personalised content. Our Bluetooth study also provided evidence that this was an accessible method of supporting personalisation: all but one of study participants had a mobile phone, and all users were successful in using a phone (either their own or one loaned to them) in order to request content.

Our scenarios each demonstrated how a viewer's personal data can be used to support the display of personalised content: Laney's recent web browsing history, Dr Jones' slide application, Mike's personal images and Ike's design inspiration. Our Bluetooth device name probe also highlighted security and privacy issues. In order to use the system, viewers were required to send a request encapsulating their interests using their Bluetooth device. User requests were broadcast in the clear, allowing any other Bluetooth device in range to eavesdrop on requests. Some similar systems required users to encode their request ahead of time reducing this privacy violation [MKHS08] but requiring users to plan ahead rather than supporting spontaneous use. Whilst our users did not pick up on the privacy issues, the use of Bluetooth itself did raise concerns for one study participant. Their concern centred around the security of their device – that the device may be hijacked over Bluetooth. Whilst security is not addressed in our scenarios, the virtualisation used in our second probe can allow each user to execute their own choice of applications without impacting other users—effectively sandboxing each appropriating user. This also has the added benefit of providing some security guarantees for the display owner.

Focussing specifically on privacy, in Section 3.2.1 we identified two broad areas of privacy concern: *disclosure through the construction of tracks and profiles* and *disclosure through the display of inappropriate or revealing content*. In addition to these risks to the user for whom content is being personalised, we also note that, in communication-focussed personalisation (e.g. instant messaging, viewing email), there are also disclosure risks for the content source.

Another challenge in personalised display networks, is the provision for supporting multiple users simultaneously. Existing research [AMS12] identifies three common techniques for showing multiple content items at a display: *space-multiplexing*, *time-multiplexing* and *integration*.

In order for display appropriation to be successful, display owners must allow their deployed displays to be appropriated. In motivation for this, we note that personalisation could reduce the burden of content production highlighted by Storz et al. [SFD+06b] and Taylor and Cheverst [TC09]. We see evidence of display owners using other peoples content for this purpose in our e-Channels study in which over 75% of display owners selected unknown content produced by others to be shown at their display. However, we also note that allowing display personalisation reduces the degree of control a display owner may have over exactly what shows on the display. Depending on the degree to which a display owner constrains personalisation, there is a risk that advertising revenues may be reduced and, on occasions, some inappropriate content may be shown. This later risk was identified by one display owner during our Bluetooth device name probe, who contacted us to ask about the measures that had been put in place to prevent explicit or inappropriate content from being shown during unsupervised periods. Despite the risks, we believe that personalisation will have a key role in future pervasive display networks, allowing display owners to offer a service to space users that encourages them to enter into, and prolong their stay in, a given space (cf. WiFi provision today).

Another key stakeholder is that of the content provider. Current display deployments are often heavily dominated by images (as seen in our own e-Channels analysis). However, the introduction of viewer-customisable rich media types into large-scale real-world deployments will provide significant opportunity for the development of a wide-range of display applications. We anticipate that content provision will become interesting, not only to large corporations (e.g. for marketing purposes), but also to small emerging companies and individual developers (similar to the way smartphone and tablet development has progressed over recent years).

Finally, an often-neglected stakeholder in display systems is the passer-by. In a display network in which content can be sourced from many places, including the viewers themselves, interpreting content could become extremely difficult. Many people rely on the information they see on digital signage to help them navigate an unknown location, provide public transport information, and inform their purchase choices. Ensuring viewers and passers-by place appropriate levels of trust in the content they see is therefore extremely important. Our e-Channels probe shows that with the right cues (e.g. branding, attribution), content interpretation can be very accurate. However when these cues are not present viewers can struggle to interpret the content and this may result in invalid trust assumptions.

In this analysis we have identified future usage models for display personalisation, exploring the incentives for a range of key stakeholders. We suggest that there is a case for display personalisation that is both technically viable and likely to be acceptable to display owners, viewers, passers-by and content providers.

3.4.2 Requirements

Collecting together our analysis and experiences from the scenarios and probes, we have distilled the following list of requirements for appropriation support in open pervasive display networks.

R1 Support for day-to-day digital signage functionality.

Whilst the focus of this work is on appropriation and personalisation support, viewer-driven content must coexist with standard digital signage content. Our previous experience with e-Campus and e-Channels has highlighted the need to ensure continued support for a minimum set of operations as follows:

R1.1 Playback support for a range of common media types (images, videos, web pages).

R1.2 Date-, time- and priority-based scheduling.

R1.3 Power-management of displays (i.e. to reduce power consumption when content is not scheduled to display).

R2 Compatibility with existing systems and connection points for third-party systems.

Significant numbers of digital displays have already been deployed, and a wealth of software tools exist to support them. The vision for open display networks aims to encompass existing entities as well as new ones. Furthermore, our own e-Channel system is popular with its current user base and servers its purpose well. For this reason, our architecture for appropriation of displays must provide connection points for existing and third party systems.

R3 Allow for user detection, request submission, and interaction through a range of existing and emerging hardware.

Mobile devices are arguably the most dominant method for supporting personalisation of current display systems. Therefore:

R3.1 The architecture should support user detection and request submission though contemporary mobile phone technologies.

Despite mobile devices dominance in current systems, sensing and interaction techniques in future display networks are unknown. We therefore suggest that:

R3.2 Our architecture for display appropriation should anticipate future changes in hardware and be designed to support a heterogeneous set of sensing and interaction methods.

More generally:

R3.3 Our entire pervasive display architecture should be open to a range of current and emerging hardware types, operating systems and platforms (i.e. not just sensing technologies, but also display hardware and network protocols).

R4 Support a range of appropriation usage models.

We have identified three distinct usage models for appropriation of pervasive digital displays. Each model has its benefits for different use cases, and for this reason:

R4.1 Our architecture should provide support for *walk-by*, *active* and *longitudinal*, appropriation.

In addition, we also highlight the likely differences in duration of the difference models, and note cases in which interrupting a viewer's personalised content may be more or less disruptive. Therefore:

R4.2 Our display architecture should provide mechanisms by which flexible scheduling (duration, interruptability constraints) can be supported.

Finally, the presentation of multiple viewers at the display means that multiple personalisation requests may be received simultaneously.

R4.3 Our architecture should consider mechanisms to support simultaneous personalisation requests from multiple viewers.

R5 Minimise security and privacy concerns.

Personalisation comes in direct conflict with privacy. Limiting unnecessary disclosure of personal data is key to maintaining a system that both supports personalisation and remains attractive to potential users.

Within the architecture, our main aim is to design a system that enables display personalisation while limiting disclosure of personally identifiable information (i.e. profiles) to the display infrastructure. We also plan to identify mechanisms to minimise the risk of disclosing information through the content shown at the display itself.

R6 Support for a wide range of applications executing in different locations and from multiple content sources.

Our scenarios demonstrated a range of possible applications for display personalisation, whilst our Bluetooth probe showed that the likely set of applications is still uncertain. So-called 'killer applications' often emerge over time. For this reason:

R6.1 Our architecture should be open to range of (currently unknown) application types

Applications may run in a variety of locations. Therefore:

R6.2 The architecture should support the execution of local applications (e.g. to reduce latency) and also applications executing somewhere in the wider network (e.g. to leverage cloud technologies).

Furthermore, each display will want to be capable of running many applications, most likely acquired from a range of different sources. Therefore this behaviour must also be supported by our architecture:

R6.3 The architecture should support the acquisition of many applications and rich media items from multiple sources at a single display.

R7 Maintain a distinction between content creators and display owners.

Experiences with e-Channels suggest that a clear distinction between content providers and display owners is beneficial to both parties. In an open network with rich media and personalisation support we think this distinction will become even more important as the potential to engage small companies and individuals in content development arises. We therefore suggest that our architecture for appropriation of pervasive display networks treats content creators and display owners as two distinct roles with interfaces tailored to each specific role.

3.5 Summary

In this chapter, we have explored the design space for viewer appropriation of pervasive displays. We described probes that demonstrate different usage models for display appropriation, including highly-constrained display personalisation using Bluetooth device names, and open, flexible display appropriation through use of virtualisation and VNC technologies.

When considering opening up pervasive display systems to user appropriation, there is significant potential for abuse. Even content that is not intended to be offensive may be inappropriate for a particular setting or simply inconsistent with a particular image a space owner or manager wishes to convey. Conversely, a viewer may put themselves or their data at risk through misplaced trust in the display infrastructure itself and in the content it displays. We have explored some of these trust issues for display owners and viewers through a number of studies and demonstrated that, in the right circumstances and with the right usage models, display owners are more willing to open up their displays to content from other sources than one might initially expect.

Based on the scenarios, probes and studies, we extracted a number of requirements for user appropriation in open pervasive display networks; these requirements are reflected in the architecture presented in the next chapter.

Chapter 4

An Architecture for Display Appropriation

4.1 Overview

In the previous chapter we identified a set of seven requirements for an architecture to support appropriation in open pervasive display networks. Our architecture should support both conventional signage (R1) and a wide range of personalised and highly-interactive novel applications (R4 and R6), whilst maintaining compatibility with existing systems (R2). In line with the vision for open display networks described in Section 1.2, the system should accept scenarios in which displays source content from many locations (R6) and function across a range of existing and future platforms (R3). Finally, to improve user experience, the architecture should maintain clearly distinct roles and interfaces for different stakeholders (R7) and should minimise security and privacy risks for those stakeholders (R5).

In order to meet the described requirements we propose an architecture composed of three key elements:

- **A multi-platform playback software system (Yarely)** capable of scheduling and rendering content to a display in response to day-to-day signage schedules and sensor-driven personalisation requests (satisfying requirements R1, R3, R6).

- **A web-based content distribution and display management service (Mercury)** that maintains a distinction between display owners and content providers (satisfying requirements R6 and R7).
- **Mobile support for viewer appropriation requests (Tacita)** through a mobile client and privacy-preserving architecture (satisfying requirements R3, R4, R5).

Together, these elements form part of a unified architecture for supporting appropriation in an open pervasive display network. This unified architecture is comprised of a *Display Segment* and a *Network Segment*. Architectural components that are co-located with the display and serve a single installation form the Display Segment, whilst components located within the network for the purpose of serving multiple displays form the Network Segment [Figure 4.1].

The Display Segment is comprised of a set of 'Display Nodes': physical display hardware (e.g. projector, LCD, speakers), processing and network interfaces to control the physical hardware. In addition, there may also be locally-connected interaction devices or context sensors.

The Network Segment is composed of a set of devices and services that support the remote manipulation and administration of displays and their content. This may include a wide-range of operations such as content creation, management, distribution and scheduling, power management and monitoring. The Network Segment is a common location for interfacing with third-party and legacy systems. Within this Ph.D., we focus specifically on the Display Segment and Network Segment services that relate to supporting appropriation in an open display network: content distribution and personalisation requests.

Both the Display and Network Segments host multiple physical devices and support a broad set of functionality. In many cases some functionality may be hosted at either the Display or the Network Segment (e.g. scheduling), with the ideal placement location emerging from a range of factors such as processing capability, network delay, and reliability.

The three key elements of our architecture for appropriation of open display networks are shown in Figure 4.2. Yarely is a Display Segment component that supports media playback and scheduling of content in response to traditional

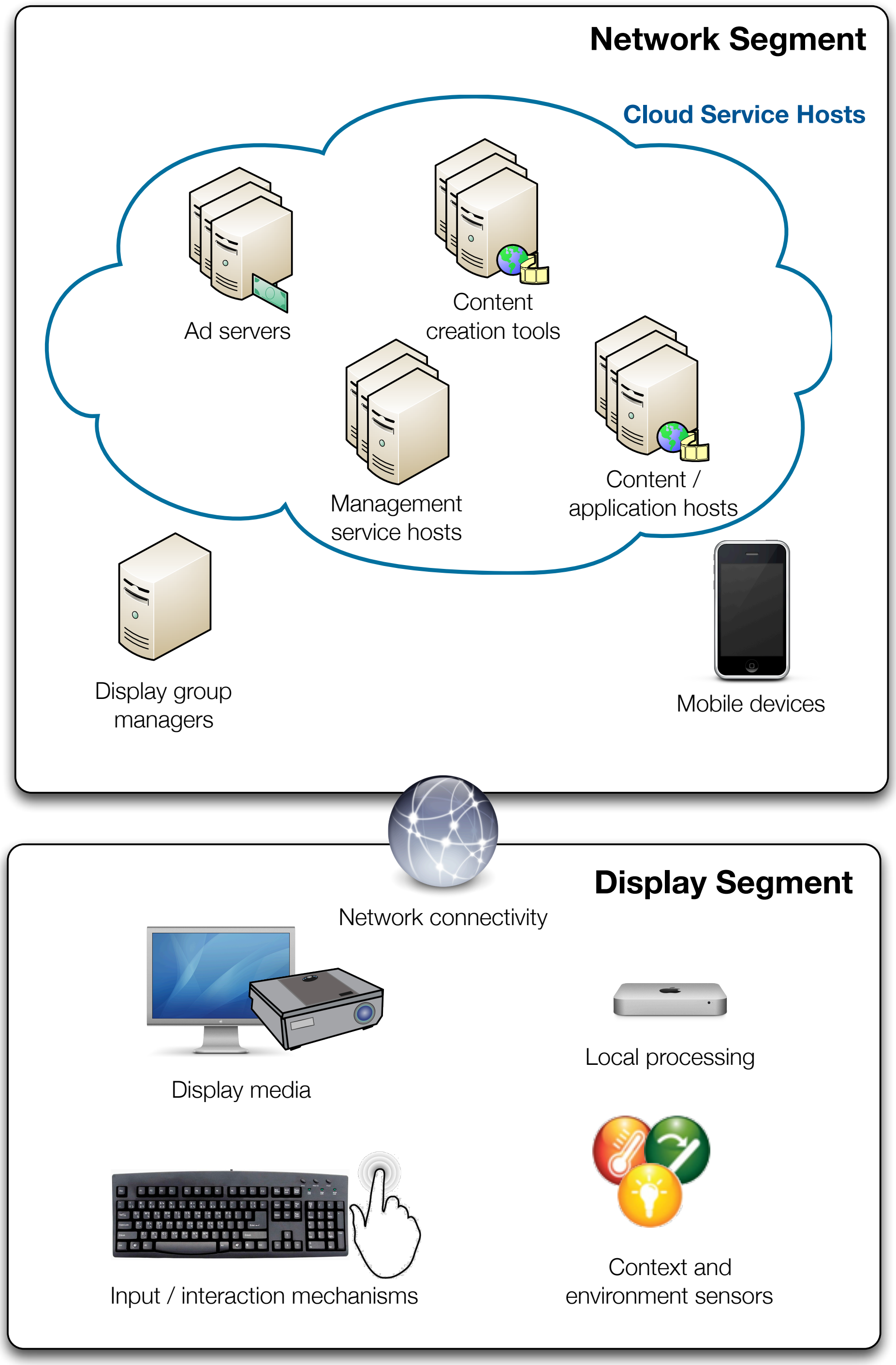


Figure 4.1: Overview of the hardware environment of a pervasive display network.

playback constraints and personalisation requests. Mercury is a Network Segment service that supports distribution of applications and other media. Communication of a content schedule from Mercury to Yarely takes the form of a content descriptor set (CDS) composed of content and associated constraints. Finally, Tacita is a Network Segment component that executes on viewers' mobile devices and allows display viewers to make appropriation requests to the displays they encounter. In the following sections, we will describe the design of each element in turn (Yarely, Content Descriptor Sets, Mercury, and then Tacita). For each element we highlight key challenges, identify core functionality, and describe the resulting design.

The work in this chapter has also been published in:

- Nigel Davies, Adrian Friday, Sarah Clinch, and Albrecht Schmidt. Challenges in developing an app store for public displays – a position paper. In *Proceedings of "Research in the Large" Workshop at Ubicomp '10*, 2010 [DFCS10].

- Sarah Clinch, Nigel Davies, Thomas Kubitza, and Albrecht Schmidt. Designing application stores for public display networks. In *Proceedings of the 2012 International Symposium on Pervasive Displays*, PerDis '12, New York, NY, USA, 2012. ACM [CDKS12].

- Sarah Clinch, Nigel Davies, Adrian Friday, and Graham Clinch. Yarely – a software player for open pervasive display networks. In *Proceedings of the 2013 International Symposium on Pervasive Displays*, PerDis '13, New York, NY, USA, 2013. ACM [CDFC13].

- Nigel Davies, Marc Langheinrich, Sarah Clinch, Adrian Friday, Ivan Elhart, Thomas Kubitza, and Bholanathsingh Surajbali. Personalisation and privacy in future pervasive display networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, New York, NY, USA, 2014. ACM [DLC+14].

A summary of this author's contributions to the above publications and the work presented in this chapter is given in Table 4.1.

Architectural Sub-System	**Author Contributions**
Mercury (Application Store)	• Concept and exploration of design considerations • Development of design into a series of UI workflows • Technology selection • Integration with other components
Content Descriptor Sets (document format for content and constraints)	• Concept and exploration of design considerations • Technology selection and development of design into document format • Representation of document format as XML Schema • Integration with other components and existing systems
Yarely (software player)	• Concept and exploration of design considerations • Design and technology selection • Majority of core (cross-platform) implementation • Integration with other components and existing systems
Tacita (mobile clients, map service and personalisable applications)	• Participation in refining an existing concept/design • Development of a number of personalisable applications • Integration with other components

Table 4.1: A summary of this author's contributions to each of the four sub-systems comprising the architecture for viewer appropriation of open public display networks. Significant contributions from other individuals include the majority of the Mercury implementation, the the initial design of Tacita, the implementation of the Tacita mobile clients, and contributions to the Yarely implementation – for details of these contributions, see the Acknowledgements (from page ii).

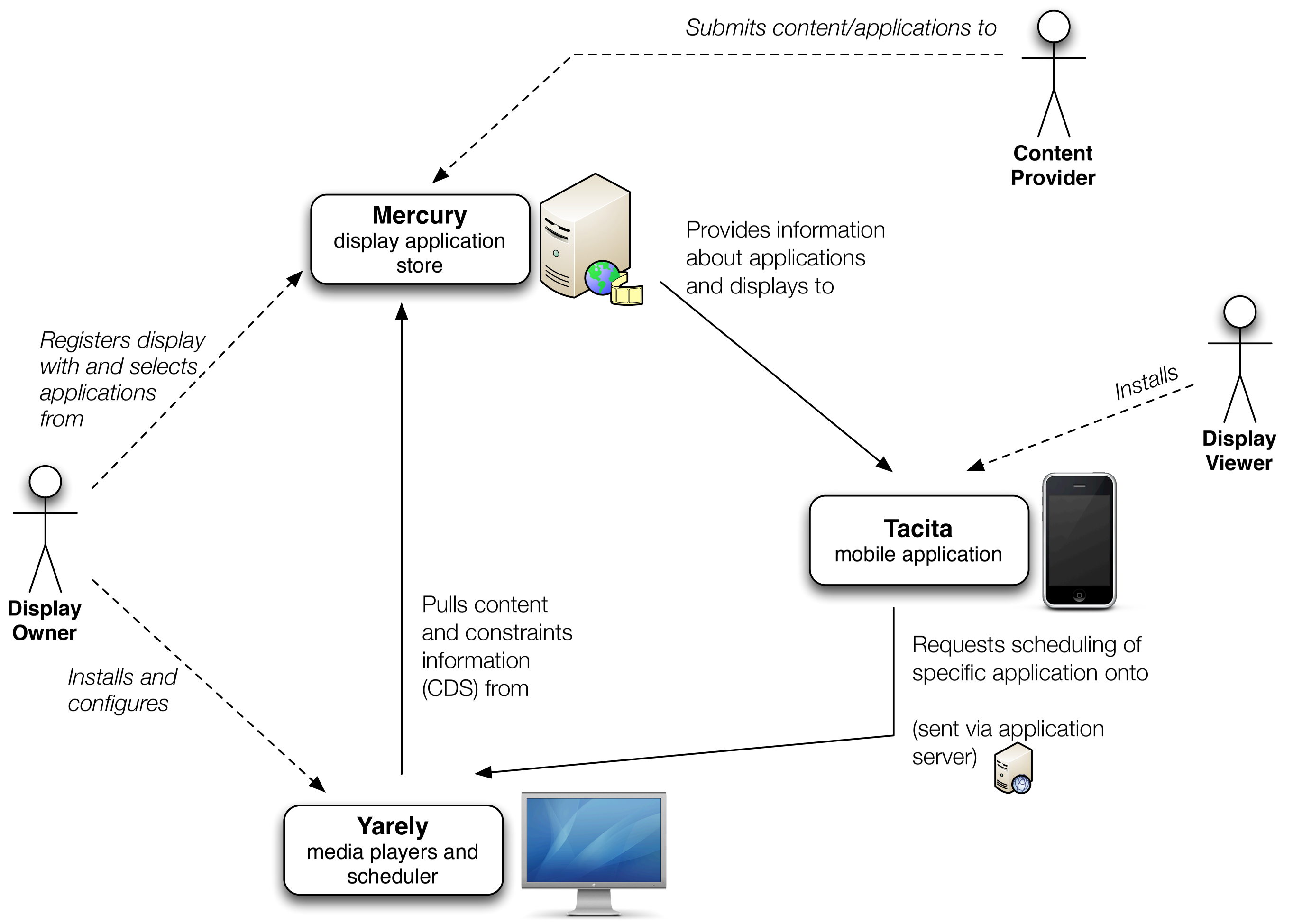

Submits content/applications to
Content Provider
Mercury
display application store
Provides information about applications and displays to
Display Viewer
Installs
Registers display with and selects applications from
Display Owner
Tacita
mobile application
Pulls content and constraints information (CDS) from
Requests scheduling of specific application onto
(sent via application server)
Installs and configures
Yarely
media players and scheduler

Figure 4.2: Architecture – high level overview.

4.2 Yarely: Content Scheduling and Playback

In the following paragraphs we describe the design of Yarely, one key portion of our architecture. Yarely is a software system designed to support media rendering and scheduling functionality. The software executes within the processing portion of the Display Segment (see Figure 4.1) and is intended to address requirements R1, R3 and R6.

4.2.1 Design Considerations

Whilst digital signage software systems are numerous, developing a signage player that supports both traditional day-to-day content and personalised rich media in a future open display network poses a number of challenges.

4.2.1.1 Addressing General Functional Requirements

Personalised, viewer-driven and interactive content must coexist alongside an established ecosystem of traditional digital signage content. For this reason, our design for a media player has a considerable focus on general signage functionality in addition to its specific support for personalised content.

Media Support
Our previous experiences with e-Channels indicate that images are currently the dominant media type for traditional digital signage content. Video and web content are also moderately popular and a small number of live streams are shown. Looking at commercial systems such as Sony Ziris [Son14] we see similar provision for images and video, plus support for standalone audio files. Support for these most common media files was therefore a key design goal for Yarely.

Scheduling
Scheduling is the process of deciding how to allocate the available screen space and time to the possible content items. The process of scheduling has two aspects:

- Physically dividing the available screen space and time. Typically this is achieved with one or both of time- and space-multiplexing, however other techniques have been proposed (e.g. integration [AMS12]).

- Comparing the current circumstances with available content items (and their recent playback history) in order to determine the most appropriate item to play on a unit of screen space/time.

Scheduling constraints are most typically described with reference to time; two approaches are commonly used. In the first approach, a timeline provides a deterministic programme for playback in which content items are listed in chronological order with a fixed start and end time. In the second, groups of items are made available for larger time-periods; scheduling is then done dynamically with a small playlist of items being generated based on all those currently appropriate to show. Whilst timeline approaches are conceptually simple, the introduction of personalised and interactive content is difficult to manage in this approach – once a schedule has been interrupted for viewer-driven content it is not immediately clear how to move back into the timeline. For example, the timeline could be resumed at the point at which it was left, but this may result in time-specific content being shown at the wrong point in the day (e.g. the lunch menu being shown some time after the café has closed). Equally, the timeline could be resumed at an offset based on the duration of viewer content but this may break sequences of content that depend on items having gone before (e.g. resuming part way through a video clip or slideshow). For this reason, in the Yarely player we focus on the second, more open, approach to time constraints.

Other design considerations for content scheduling are priority and ordering. Marking some content items as high priority can allow a display owner to respond to a temporary set of circumstances or event. Two approaches to scheduling priority content are common: content of a high priority may be shown exclusively, starving out lower-priority content; alternatively high priority content may simply be shown with greater frequency, receiving the majority of the available screen space and time but allowing other items to be displayed periodically. Content ordering allows display owners to determine which content items follow other items. Ordering content items in this way is particularly useful for the creation of slideshows of images and stories.

Power Management
Finally, both e-Campus and commercial systems (e.g. Sony Ziris [Son14]) provide mechanisms for allowing the control of physical display hardware. This allows display owners to power off displays as required, providing benefits with regard to energy consumption, light pollution, reducing cognitive load for passers-by, and reducing the burden for content creation (by limiting the number of content hours required).

4.2.1.2 Sensing and Appropriation

This thesis is centred around viewer appropriation of pervasive displays. Our requirements specify that appropriation requests may come in through a range of viewer detection, request submission, and interaction technologies based on both existing and emerging hardware.

A number of different methods are currently used for viewer detection. A display may scan for a viewer's personal mobile device (e.g. by detecting a unique network device id associated with a mobile phone) and assume that presence of the device indicates presence of the owner. Alternatively, the display may detect the viewer themselves (e.g. using a camera, perhaps with face detection), or may detect the viewer as a result of display interaction (e.g. receipt of a touch event at a touch-screen implies the presence of a viewer). Depending on the appropriation usage model being used, a personalisation request may then be generated or received. As in viewer detection, techniques and technologies used at this stage may vary including data mining from existing profiles (e.g. as supplied by some cloud service) and receipt of an explicit request sent over some short-range communication protocol amongst others. Once personalised content appears at the screen, viewers may interact with the screen – while a wealth of methods exist for display interaction (e.g. touch-screen, gaze-tracking, personal mobile device applications), no clear consensus has yet emerged.

The lack of convergence on specific technologies for viewer detection, request submission, and display interaction creates a clear challenge for supporting these functionalities within our media player. Furthermore, future detection, submission and interaction techniques cannot yet be anticipated and so the player must be open to the addition of new functionality to support these.

4.2.1.3 Openness and Extensibility

A key goal of the architecture is to support integration with future open pervasive display networks. Most current signage players assume that they are controlled by a single authoritative source that supplies them with both content and a play out schedule. By contrast, Yarely should allow connection to multiple content sources simultaneously.

This represents a fundamental shift in thinking regarding signage systems – instead of a player being treated as a simple "slave" that plays the content that is sent to it, we aim to create a more intelligent player that is able to schedule content from multiple sources that are not necessarily aware of each other and hence may provide the player with conflicting play out requirements.

Current digital signage players focus on simple media such as images, video, audio and web content. The combination of openness and developments in interaction technologies creates the potential for new media for pervasive displays including complex applications. The previous section identified a need for extensibility in order to support a wide-range of sensing and appropriation mechanisms. Similar extensibility is required to support this range of current and future display application types. More generally, aiming for extensibility in all portions of the Yarely player (e.g. networking, scheduling) provides a mechanism for constant adaptation with minimal changes to the core player.

Finally, our desire for openness also stretches to open technologies and support for integration with existing and future third party code. Mechanisms should be provided for components developed by third parties to interact with our player and for our player to integrate with existing systems. In order to best support a wide-range of systems we propose a multi-platform design and suggest that communication between components (both within the player and between the player and other portions of the architecture) should be in a clearly-defined, open format, allowing easy integration of third-party components.

4.2.1.4 Resilience and Disconnected Operation

In order to have our player widely adopted it is crucial to develop a reliable solution that will scale to very large deployments.

Two approaches to digital signage software design are common. In the first, a thin client is connected to a remote server. The remote server manages the client by creating a schedule and processing that schedule to identify the current item to be played – graphical data for that item is transferred to the client which renders the data to display the item. Web-based and cloud-based systems are increasingly popular examples of such remotely managed systems (e.g. signagelive [Rem12]). By contrast, in the second method, a largely stand-alone player manages its own content schedule and uses this to identify the item to be played. All associated processing is completed local to the display. Hybrid approaches also exist in which some processing may be executed locally whilst some is also offloaded to a central server.

Centralised approaches to digital signage can allow quick and cheap deployments but can leave the screens vulnerable to failures – should the network or server software become unavailable the clients will fail. Recent developments in HTML 5 APIs [W3C08] allow for offline operation but require a modern browser and sufficient local storage for offline applications to maintain local state whilst disconnected. Hybrid approaches can reduce the effect of a network or server problem but centralised aspects remain vulnerable – for example, our experiences with the centralised scheduler in the e-Campus deployment showed that network and database bottlenecks resulted in slow content transitions and frequent periods of down-time. For this reason, we favour a 'thick client' design in which display nodes can operate independently for extended periods of time. Specifically, we suggest locating a scheduling component at the display itself.

In an open display network in which content comes from multiple sources, this itself minimises some effects of centralisation. When content at one source is unavailable then the player can continue to show content from other sources. Furthermore, this openness also helps to support large deployments as the emergence of many content sources will help distribute requests from displays. Caching techniques can also be used to further reduce the number of requests made for content items.

4.2.2 System Design

Yarely is our media playback and scheduling component designed to execute at display nodes.

We selected a component-based approach for Yarely. Use of components allows easy adaptation and integration of new components as new hardware develops.

To support both day-to-day signage and complex media including personalised and sensor-driven content we identify five core operations, each represented by a separate component. Each of the five resulting components communicates with the others through a shared event channel – this message channel also helps to allow extensibility and openness as it allows the development of new components that need only access the event channel to communicate with existing portions of the software. The five core components and central event channel are shown in Figure 4.3.

The five components are as follows:

Subscription Management

The Subscription Management component is responsible for maintaining connections with content sources for the purpose of requesting and receiving the Content Descriptor Sets that describe content subscriptions (the design of these Content Descriptor Sets is described fully in Section 4.3). The Subscription Management component polls periodically for changes in its subscriptions and passes any changes on to other components (e.g. the Playlist Generator and Scheduler).

Sensor Management

The Sensor Management component reads data from sensors and environmental sources (e.g. context, user presence and interaction) and passes received events and data to other components. For example, such events may be used to trigger the injection of interactive content onto the node.

Playlist Generation and Scheduling

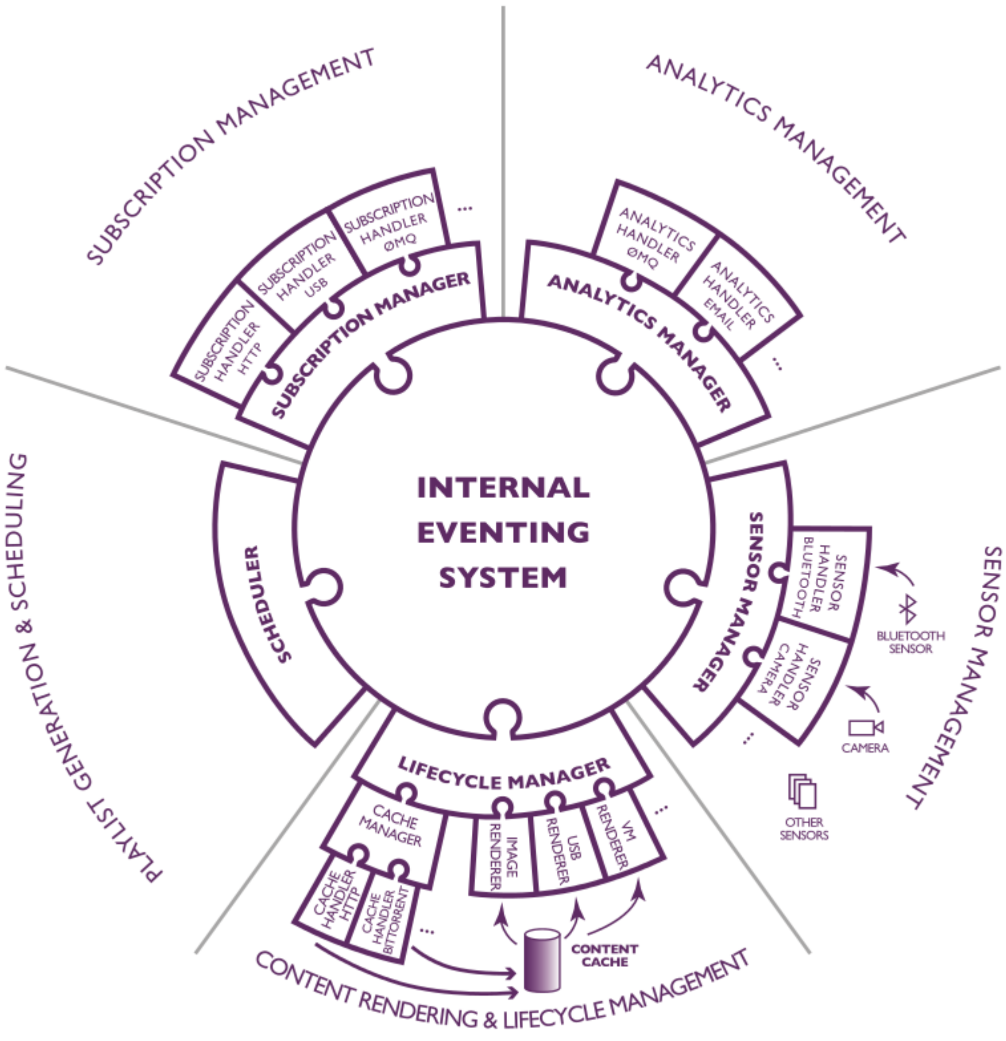


Figure 4.3: The Yarely software design. Five core components communicate through a central event channel.

This module processes the available subscriptions to derive a playlist of content items to be played in the immediate future and is responsible for the scheduling of these content items onto the screen in response to changing circumstances (e.g. time, sensing events) and in line with the constraints specified in the Content Descriptor Set.

Content Rendering and Lifecycle Management

Content Rendering refers to the physical presentation of content items onto output hardware (e.g. display, speakers). The Lifecycle of such content items includes the caching of media at the display node, the preparation of content in advance of playback, playback, and cleanup on completed playback.

Analytics

The Analytics component is responsible for returning data to external servers (e.g. the Descriptor Factory or application provider). These may take the form of summary statistics or notifications on a per-event basis.

With the exception of the Playlist Generation and Scheduling component, each of the components is comprised of a managing element (Manager) plus a set of associated plug-ins (Handlers). Each Manager provides an interface for other elements within the system that allows them to access its functionality. The Handlers each address the specific software or hardware requirements of providing that functionality. For example, in the case of Subscription Management, the Subscription Manager controls the retrieval of subscriptions, and shares the result with other components, but the actual retrieval of the Content Descriptor Sets that comprise the subscription is delegated to a set of Handlers, each concerned with a specific method of subscription retrieval (e.g. reading from the file system, fetching over HTTP). This design is intended to allow easy coverage of a wide-range of networking protocols, sensing technologies, media formats etc. – each new format that requires support can be provided through implementation of a dedicated Handler. Furthermore, the approach can also be used to improve reliability by allowing a single content source to make their files available over multiple protocols concurrently, ensuring that even when one handler fails to fetch a file the item may still be fetched by another component.

4.2.3 Comparison with Prior Work

Yarely follows a thick-client design in which the display node handles not only rendering of a media item onto the physical display device, but also aspects such as subscription parsing, scheduling, and analytics gathering. This design decision is intended to allow the nodes to be open to input from multiple sources and to support continued operation during periods of network interruption.

By comparison, much of the prior work has tended towards thin display clients that provide rendering of content selected by a remote scheduling service. For example, in FLUMP [FWDF96] content selection was made at a web server based on the presence or absence of active badges. Each display node executed only a traditional web browser which showed the output of the web server as a rendered HTML file. This web-based approach was also used in Digifieds – the Digifieds web server maintained sets of classifieds for each display location and deployed displays could use a web browser to show the resulting output. Recent developments in HTML 5 [W3C08] can allow these thin clients onto web content to continue displaying content despite network disconnectivity. Other non-web-based systems have also tended towards lightweight display clients whose content schedule is managed by a server, for example, in Greenberg and Rounding's Notification Collage [GR01], a central server accepted notification events from a range of posting clients (e.g. a video generator). Client applications at the displays would receive new notifications from the server and display the resulting media. Some control of the display was however retained at the client – viewers could alter the placement of media items shown at the display by hiding or moving the associated components.

Perhaps the closest design to that used in Yarely was the e-Campus system, in which each display node executed a local scheduler that could arbitrate between conflicting content requests. Any number of local or remote applications could make API calls to the scheduler asking for an item to be played and the local display scheduler would use priority and transaction logic to decide which items should be shown. Acceptable input for the display scheduler included single content items or sequences to be played immediately – priority comparisons were then made between available items and an existing content in playback was preempted as required. By comparison, Yarely accepts a complex schedule description comprising of multiple items that should only be played if the local scheduler can determine specific constraints have been met. This allows the Yarely scheduler to derive a playlist ahead of time and handle time and date constraints as well as priority and ordering. Furthermore, whilst e-Campus was theoretically able to make local scheduling decisions, day-to-day signage content was controlled by a central server application, e-Channels, that made scheduling decisions for each of the networked displays and then sent the appropriate requests on an item-by-item basis to the individual display nodes.

In line with the general functional requirements identified in Section 4.2.1.1, Yarely's support for media files remains similar to that provided by similar systems (e.g. e-Campus [SFD06a, SFD+06b], Sony Ziris [Son14], Xibo [GHtXP14]). Furthermore, the component-based design allows easy addition of new rendering components.

4.3 Content Descriptor Sets: Describing Content and Constraints

The previous section described the design of Yarely, our media player and scheduler designed to execute on display nodes. In Section 4.4 we will describe a service for maintaining a repository of applications to play at those nodes. In this section, we focus on the communication mechanism between these two architectural components, i.e. Content Descriptor Sets (CDSs).

The CDS is intended to be an open format that can describe a range of content and constraint types (e.g. day-to-day signage (R1) and novel applications (R6)). More critically, as the primary method of communicating with Yarely, the CDS is a key integration point for existing and third-party systems (R2). The CDS is transferred from components in the Network Segment to the Display Segment.

4.3.1 Design Considerations

4.3.1.1 Representation of a Range of Media Items

Section 4.2.1.1 identified a number of target media types for the Yarely media player: images, video, web (i.e. HTML), and audio. Our experiences with cloudlets and virtualisation also suggest a need to represent more complex content and applications.

In order to be able to support distribution of these different forms of content to the display nodes, the Content Descriptor Set should be able to describe a source for acquiring the content files, plus some mechanisms for ensuring the correct file has been received (e.g. a file hash and content type). Whilst most media files would typically be represented by a single file (perhaps replicated in multiple locations), our more complex content prompts a need to support representation of single content items comprised of multiple files (for example, the virtual machines used by Elijah were comprised of an overlay descriptor file, a base VM, an optional set of disk image files and an optional set of memory files).

4.3.1.2 Support for a Wide Range of Scheduling Constraints

Day-to-day digital signage typically operates on date and time-based schedules that may take the form of very fixed timelines or more loose requirements that particular items should play in the morning, on a specific date, or only at weekends. Ordering constraints are also commonplace, determining that a specific group of items should be played sequentially.

In addition to these traditional signage constraint types, a range of contextual constraints may also be required in future systems. For example, data from local sensors could provide a scheduler with information about its environment (e.g. noise levels, temperature, number of faces detected around the display). Such context could be used to schedule specific content items in response to particular thresholds being met (e.g. only showing an advertisement when twenty or more viewers are present to see it, or to restrict playing of an audio file to times when noise levels are low). Due to the unpredictable nature of these triggers, the Content Descriptor Set format should encourage the creation of flexible schedules in which multiple items could potentially be available for playback at any time – this is a stark contrast to timeline based systems in which schedules are generally fixed with only limited opportunity for branches to represent conditional scheduling.

4.3.1.3 Support for Display Appropriation

Our focus on display appropriation highlights the importance of a specific set of environmental triggers related to the presence of individuals with appropriation requests. These triggers may take a range of forms, for example the presence of a Bluetooth device name matching a particular format (as in Davies et al. [DFN+09a]), recognition of an RFID or other short-range communication device (as in FLUMP [FWDF96]), or a third-party authentication service (as in Prospero [Con07]).

Like other contextual triggers, support for display appropriation is dependent on flexible schedules that can cope with a user arriving at the display and interrupting the current program. This also suggests some provision for determining which items can be interrupted and by what – for example, including a mechanism for priority specification could allow some items to be explicitly flagged as not suitable for interruption by viewer content, whilst other items might allow it.

4.3.1.4 Openness, Extensibility and Compatibility

We have previously highlighted the need for our media player to ensure compliance with future open pervasive display networks including new content sources, media types and signage software. These requirements also have implications for our communication between content sources and the media player.

The purpose of the CDS is to describe content and constraints for interpretation by the display nodes. As content and sensing technologies change, the CDS will need to be able to describe these new media and inputs. Providing an extensible format will allow adaptation to new needs whilst maintaining backwards compatibility.

Furthermore, in order to allow openness with the widest range of existing and future systems, the format for description should in itself be open and based on a widely-supported format.

4.3.2 System Design

A Content Descriptor Set (CDS) is a method of describing content availability. Content Descriptor Sets are designed to be transferred from a content source to a display node's media player. This is represented by the line from Yarely to Mercury in Figure 4.2. More generally, we conceive a model in which CDSs may be transferred between these two components either through a pull method (as pictured in Figure 4.2) or by pushing from the source to the display node. Our CDS is intended as a descriptive document designed to be transferred between content source and media player and is agnostic of the underlying transport protocols.

Whilst in this thesis we focus predominantly on Mercury as the source of content for playback by Yarely, we consider all entities capable of producing a CDS as *Content Descriptor Factories*. We consider that each Content Descriptor Factory may provide CDSs for multiple display nodes, and that each display node may pull CDSs from many Content Descriptor Factories. A variety of transfer patterns for Content Descriptor Sets is show in Figure 4.4.

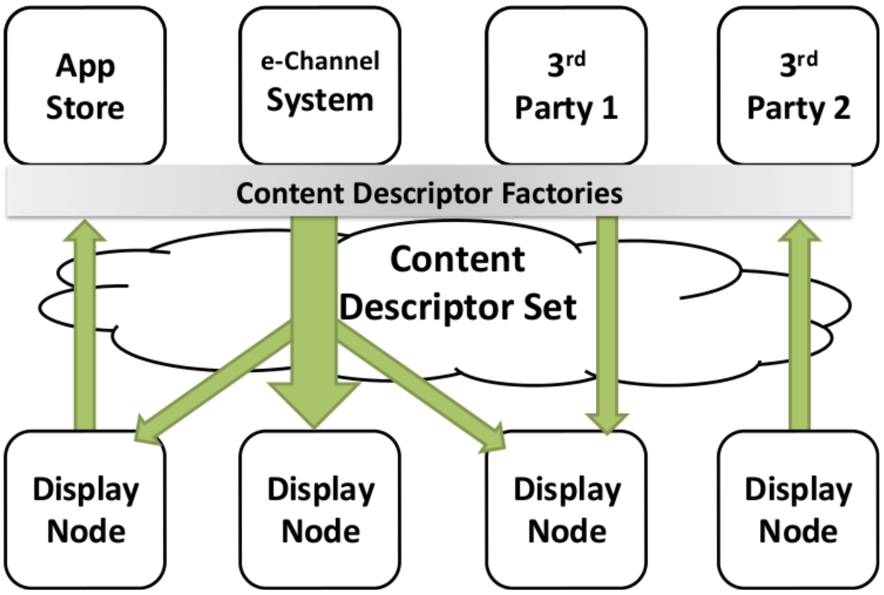


Figure 4.4: Content Descriptor Set Transfer: Pull and Push

A CDS describes both the media to be played at a display (i.e. location, file type, file size) and the circumstances in which it should be played (i.e. the scheduling constraints described in Section 4.2.1.1). Groups of media items should also be represented (e.g. slideshows, items from a single source). We therefore identify three key entities to be described within a CDS: constraints, content items and content sets.

A *constraint* describes a limitation regarding the circumstances in which something (a content set or content item) may be shown at a display. For day-to-day signage purposes, typical constraints include those based around time, priority and ordering. For viewer- or environment-driven systems, constraints might relate to data matches on a particular group of sensors (e.g. temperature is 30℃, audience size is greater than ten).

A *content item* represents the set of data that is required to represent a single media item or application. A content item may include a direct representation of the data (e.g. text to show on a scrolling display), or more likely a set of pointers (i.e. URIs) that describe where to retrieve the data from (e.g. a path to a local video file, a URL for an application hosted on the web). Depending on the media or application format, this may be a single block of data (or file) or multiple blocks. For example, an image is typically represented by a single data file, but a virtual machine may be comprised of multiple files (one for hard disk storage, one for memory, one for configuration). Where a URI is used to describe the location of a media file, the content item representation should also include the final file size and a hash of the data to allow playback software to confirm that it has fetched the correct file.

In addition to representation of the data for the media itself, the content item representation should also provide the MIME type of the content items in order to inform playback software about the kind of software requirements needed to show the item on a display.

Finally, a *content set* represents multiple content items that have a specific relationship to each other (e.g. ordering), or can be referred to as a single unit (e.g. for the purpose of applying scheduling constraints). For example, a simple signage display showing a series of slides in order could have those slides represented as an ordered content set.

The Recursive CDS Structure
A key focus of the CDS is its focus on supporting openness in future display networks, and specifically the sourcing of content items within such networks. If future displays are "open to applications and content from many sources" [DLJS12, p. 9], then the role of the CDS is to represent not only the set of content provided by a single display source, but also to represent the results of merging content from multiple sources either at the display node itself or at intermediary services within the network.

Given a dual role of representing both the set of content provided by a single display source and a complete set of content to be accepted by a display, the CDS design provides a simple mechanism for merging content sets. This is supported through a recursive structure in which content sets may themselves contain other content sets. This recursive structure is shown in the UML class diagram contained in Figure 4.5.

4.3.3 Comparison with Prior Work

As described in Section 4.2.3, many client-server digital signage systems have the display node's delegate scheduling to the server such that the server generates events or messages to the client at regular intervals, each describing a single content item to be rendered (as per the Elvin events used in e-Campus [SFD06a]). Variations of this pattern can also be seen in web-based approaches in which the display node renders a single page that in turn received regular DOM updates to alter page content (e.g. over Ajax).

By contrast, the CDS document passed from Content Descriptor Factory to display node completely describes all content available for that display from that particular source and the associated constraints. The problem of describing media presentations (and, more generally, multimedia documents and authoring systems) has been the subject of extensive research for more that two decades (e.g. Boll et al. [BKW00]). Of most relevance to our work is SMIL [W3C03]. Like the CDS format, SMIL is an XML-based file format that describes both media and scheduling information. Both the CDS and SMIL formats handle a range of media types (video, images, audio, HTML), traditional signage constraints (se-

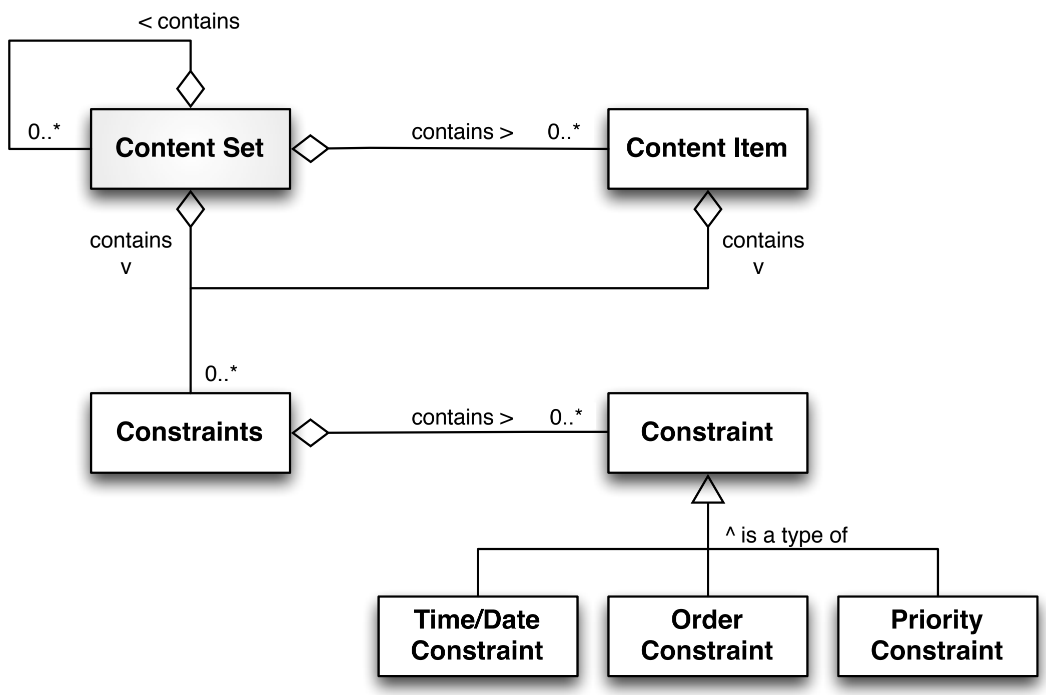


Figure 4.5: UML class diagram for the Content Descriptor Set. The recursive nature of nested content set objects is shown by the aggregation relationship in the top-left corner. Constraint subclasses shown are examples rather than an exhaustive set.

quential ordering, date and time), and interactivity in response to viewer input. The CDS format supports additional complex media types composed of multiple files – this allows new forms of media (e.g. virtual machines, applications) to be described in CDS documents and helps to the architecture to address the requirement to support a wide range of applications [R6.1]. By contrast, SMIL provides additional support for screen multiplexing (theoretically possible with a CDS but not defined as part of the schema definition). Both the CDS and SMIL document formats allow nesting – a CDS document can contain references to other CDS files, and a SMIL document may link to another SMIL script.

4.4 Mercury: Content Provision and The Role of Application Stores

In the mobile phone domain, application stores have been shown to provide a platform that removes many of the barriers to deploying applications onto a personal mobile device. Now the dominant method of distributing applications to smartphones and other devices, the presence of these stores allows universal access for both developers and users. Our architecture features a similar entity specifically for pervasive display applications and devices. Use of such a display application store was illustrated in the scenario provided in Section 3.2.4.

Our display application store (Mercury) is primarily designed to solve the problem of content distribution within pervasive display networks. In a similar vein to the e-Channels system described in Section 1.4.4, our application store is designed to provide mechanisms for 'Content Providers' (also referred to as 'Application Developers') to make content available through the store, and for 'Display Owners' to navigate through content and select that content for playback at their display. This corresponds to the 'content descriptor factory (CDF)' element of the overall architecture's computational model and resides on the 'management service hosts' hardware.

A secondary goal of the store is to provide support for Display Owners to maintain a catalogue of their own displays. This helps to support our primary functionality but also allows Mercury to maintain, and share, a directory of all registered displays. This functionality corresponds to the 'map provider' element of the overall architecture's computational model. Likewise, the internal directory of applications can also be exported by the application store, corresponding to the 'service directory' element of the computational model. Finally, the application store will maintain some data about content selection and playback, corresponding to the 'analytics' element of the computational model.

4.4.1 Design Considerations

Whilst the overall concept for Mercury is similar to our existing understanding of application stores, the translation to pervasive displays brings a number of new challenges.

4.4.1.1 Stakeholders

Pervasive display networks feature a complex stakeholder set including display and space owners, viewers, passers-by, content (application) providers and ad brokers. An application store has the potential to engage at least three of these stakeholders: display owners and viewers may use the store to discover and select content for viewing at a display, whilst an application provider will use the store to disseminate their applications/content to a display [Figure 4.6].

The existence of these many interests adds a layer of complexity to applications as they emerge to benefit different stakeholders. For example, some applications may be designed with primary benefits to the display or space owner (e.g. advertising content with financial benefit for the display owner, information delivery that helps a space achieve its purpose effectively, or jukebox applications that improve the environment and draw people into the space). Equally, there may be a set of applications that primarily benefit passers-by and display viewers (e.g. those that display information of personal interest or allow an individual to complete a specific task – see also Section 3.2). Whilst both application types have a primary beneficiary, each also offers benefits to other stakeholders. For example, display owners may find attention transfers from viewer content to other items on the display and application developers may gain a number of benefits from their application being shown (e.g. improved reputation, financial payment,

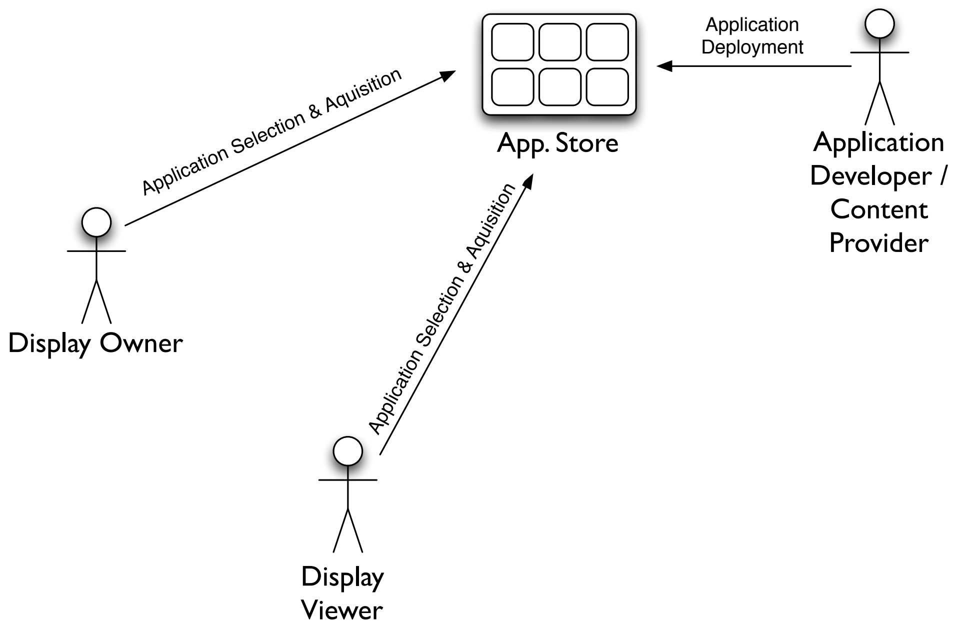

Application Deployment
App. Store
Application Developer / Content Provider
Application Selection & Aquisition
Display Owner
Application Selection & Aquisition
Display Viewer

Figure 4.6: Application store for pervasive displays: stakeholder overview.

data collection).

For the purposes of this design, we have decided to focus on provision for two key stakeholders: display owners and content/application providers. Our design does not feature an explicit graphical interface for content browsing and selection by display viewers, but instead the design allows for flexible access to the underlying data set such that this functionality can be provided by other systems (e.g. Tacita, Section 4.5) and through future extensions of the application store.

4.4.1.2 Business Models and Purchasing

Given its complex stakeholder set, it is unsurprising that the business models underpinning an application store for public displays must also differ from those associated with similar stores for applications on personal mobile devices. The promotion of interactive applications and appropriation support also means that current business models for digital signage are also a poor fit.

Current business models for display networks are often advertising-focussed, with companies maintaining large networks and selling the space to those wishing to advertise (e.g. DSN [Dig12]). Müller and Krüger [MK07a] identify three options for advertising payment models on pervasive displays: per-impression, per-view and per-attendance. Per-impression models are simplest to implement with a charge being made based on the number of seconds of screen-time irrespective of the presence (or absence) of potential viewers; in a per-view based model the advertiser would pay for each person who has seen the advertisement and in per-attendance the advertiser pays only for those who have both seen the display and acted upon it. Within the mobile application store domain a one-off purchase model is typical (a single financial transaction allows unlimited use), whilst other mobile and cloud services typically offer a subscription model in which a purchase allows access for a specified time period or predetermined consumption level (e.g. a one-month access period, or a 50 MB data allowance).

Whilst Müller and Krüger's models provide a useful tool for calculating the cost associated with the display of content (applications) in a pervasive display network, the issue of who pays that cost is unclear. In Section 4.4.1.1 we highlighted how different applications may offer benefits to the different stakeholders, this in turn may impact how payment is exchanged.

The role of advertising in business models for display application stores poses an additional challenge. Advertising content/applications may exist as distinct entities in the store, similar to the way in which advertising content is currently produced for displays. However, the kinds of in-application advertising seen within current mobile applications may also emerge as a mechanism for application developers and other stakeholders to generate additional revenue.

Finally, the single-direction payment models associated with mobile application stores lend them well to traditional currency payments. In a display application store, the multi-directional payment possibilities (display owner to content provider, content provider to display owner, viewer to display owner etc.) mean that other currencies may emerge. For example, exchanges and credit based systems.

4.4.1.3 Scheduling

In mobile phone based application environments the decision to start an application is made by the user of the device, whilst in conventional display networks the decision to schedule content is made by the display owner. In an open display network with application stores and support for user appropriation, scheduling decisions must accommodate preferences from multiple stakeholders.

Perhaps most obviously, display owners may expect to maintain much of the control they currently have over their displays, determining which content is acceptable for their displays and the circumstances in which it should be shown. In addition, application creators may want to restrict their content to circumstances in which it will behave as intended and serve the purpose for which it was designed, resulting in restrictions to particular display configurations, locations or times of day. Furthermore, whilst content selection by viewers is not the focus of this portion of the architecture, we note that user appropriation adds additional scheduling complexity in order to meet a viewer's expectation that their personalised content will show at a display. Figure 4.7 demonstrates how a set of different constraints can emerge for display applications within the application store.

Content selection and compliance with content constraints is not a goal for the application store as this is handled by Yarely, the scheduling and media playback element (see Section 4.2). However, within the application store providing appropriate interfaces for scheduling control is an important issue that must be addressed.

4.4.1.4 Analytics

Web analytics have become a useful tool for reporting data and optimising web usage. In a similar vein, we envisage analytics for display applications emerging as a key tool to help application developers understand how stakeholders engage with applications and displays.

Using data gathered in two focus groups conducted with researchers and students developing public display applications, we identify two core data categories for display applications:

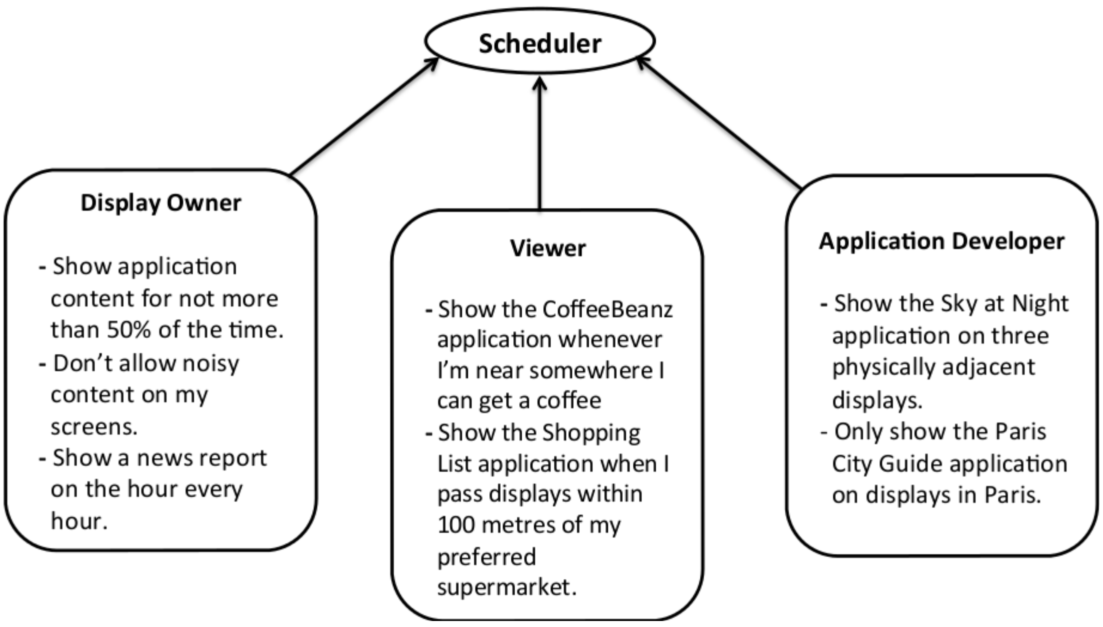

Scheduler
Display Owner
- Show application content for not more than 50% of the time.
- Don't allow noisy content on my screens.
- Show a news report on the hour every hour.
Viewer
- Show the CoffeeBeanz application whenever I'm near somewhere I can get a coffee
- Show the Shopping List application when I pass displays within 100 metres of my preferred supermarket.
Application Developer
- Show the Sky at Night application on three physically adjacent displays.
- Only show the Paris City Guide application on displays in Paris.

Figure 4.7: Application store for pervasive displays: scheduling conflicts (adapted from Clinch et al. [CDKS12]).

Execution Data data that describes the system elements of the execution environment and performance of an application. For example, the number of distinct displays the application is shown on, the geographic location of displays, the hardware and software environment at a display, applications scheduled before or after this application, scheduling conflicts and details of application crashes/failures.

Interaction Data including contextual and demographic data about human traffic flow, implicit interactions with the application (e.g. presence, attention changes) and explicit interactions with the application. Explicit interaction data ranges from simple usage patterns (e.g. times the application is selected/launched), interaction patterns within the application (e.g. length of gesture sequences, scrolling events) and intentional reporting about the application (e.g. ratings, reviews).

Our focus groups suggested that analytics data offers most benefits for optimising application functionality (e.g. bug fixes, taking advantage of new hardware trends) and exposure (e.g. altering the usage demographic, publicising positive reviews).

We note a key tension between analytics provision and our requirement for measures to minimise privacy concerns (R7). Typical techniques to address this include data anonymisation and aggregation.

4.4.2 System Design

The application store (Mercury) is intended to act as a content distribution and display management service. Content providers can use the store to share applications within the display network. Display owners can use the application store to browse available applications, transfer selected applications to displays and manage display schedules.

Our design for Mercury incorporates two interfaces: a user-interface designed as a set of web pages, and a set of RESTful APIs that can be accessed outside of those pages to allow compatibility with other software. This architecture is represented in Figure 4.8. Both interfaces utilise the same data storage model with twelve key entities [Figure 4.9]:

Application
: An entity that represents media designed to be shown on a display. An Application is produced by a Development Company.

Billing Model
: A description of a payment model by which an Application can be purchased, including cost (e.g. a one-off purchase at a cost of £5, a subscription at a cost of £2 per month, or a view-based model at a cost of £0.50 per view).

Category
: A thematic group of Applications (e.g. games).

Developer Profile
: Represents a User's membership of a Development Company.

Development Company
: Represents the entity that contributes Applications to the store.

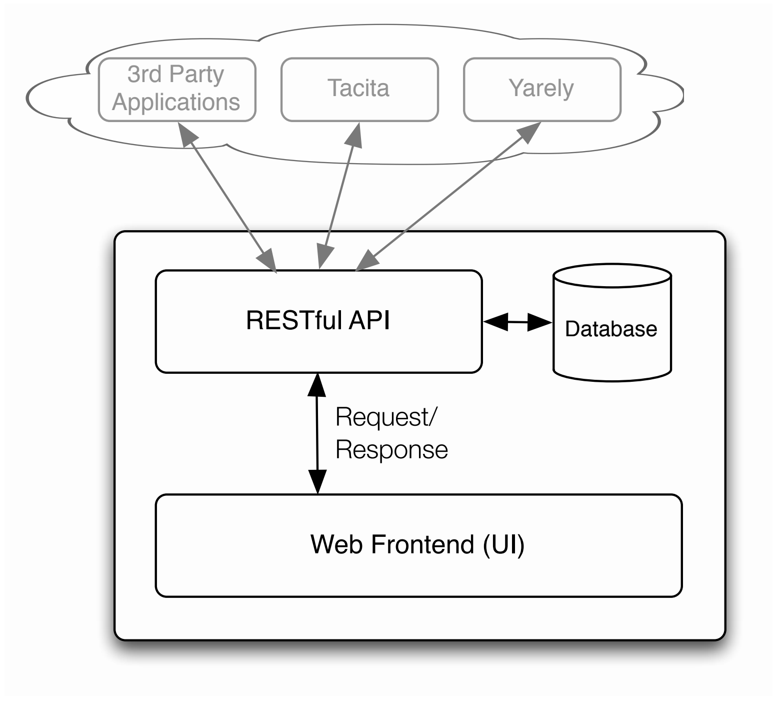


Figure 4.8: Application store for pervasive displays: architectural overview. A RESTful application programming interface (API) serves both the Mercury web-frontend and other portions of the overall architecture such as Yarely and Tacita.

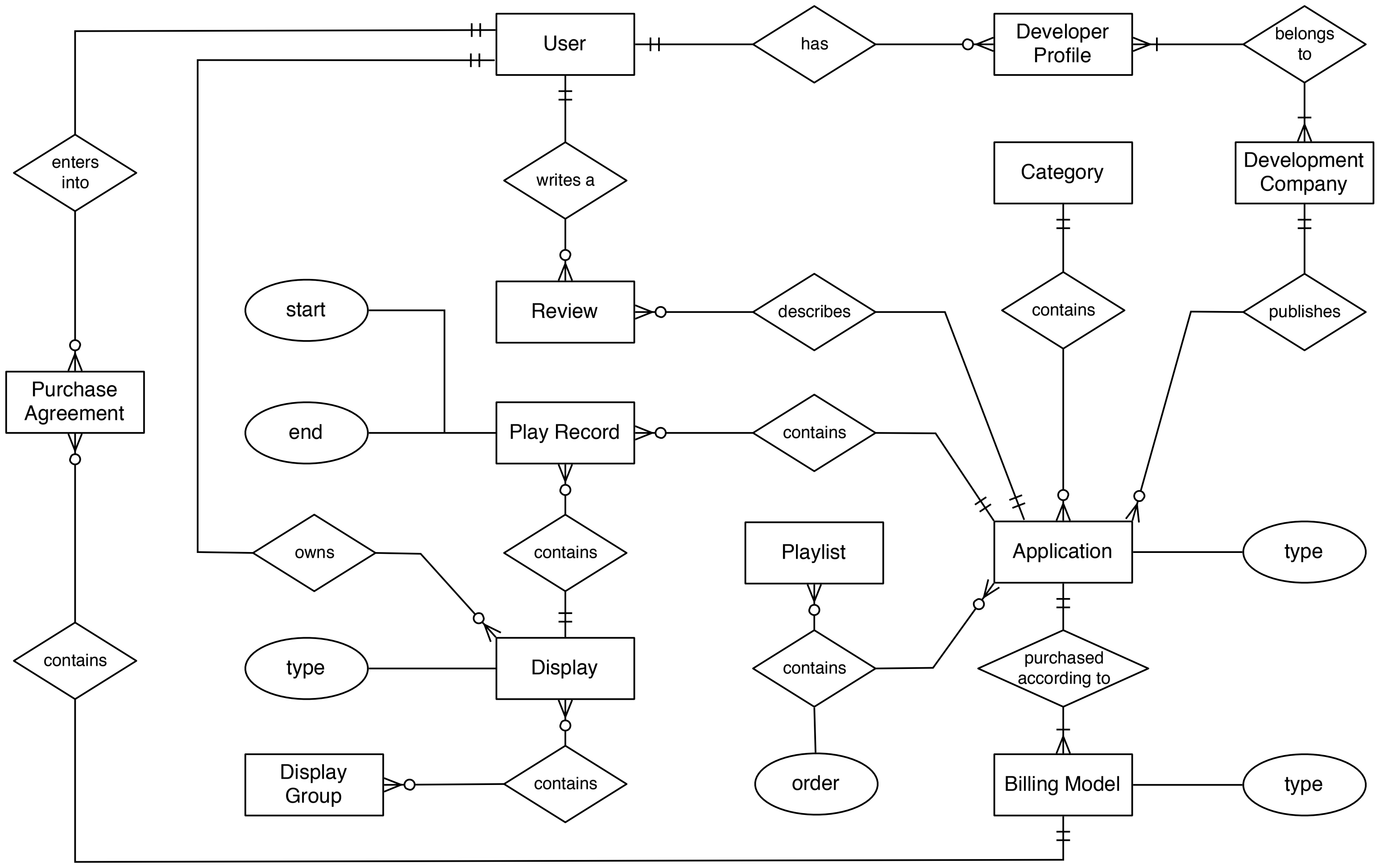


Figure 4.9: Application store for pervasive displays: entity-relations.

Display

A representation of a Display Node within the physical network.

Display Group

An unordered collection of Displays.

Play Record

A description of the circumstances in which an Application has been played at a Display (e.g. date, time, duration).

Playlist

An ordered collection of Applications.

Purchase Agreement

An entity that represents a User purchasing an Application according to a specific Billing Model.

Review

A critical evaluation of an Application written by a User who has previously purchased that application. A review may consist of text and/or a numeric rating.

User

An individual who is registered to use the application store.

4.4.2.1 Web Interface

The web user interface will be the primary interface for display owners and content providers. To design the interface we generated a basic design composed of a sidebar, top menu and main frame. These designs were initially sketched out in order to allow rapid evaluation [Figure 4.10].

A more extensive set of sketches were then developed around a set of workflows. Each workflow represented the process of completing a core operation using the application store user interface. Our workflows included actions carried out by a display owner (e.g. selecting content for a display, creating playlists, managing displays) and also those executed by application developers (e.g. adding a new application to the store).

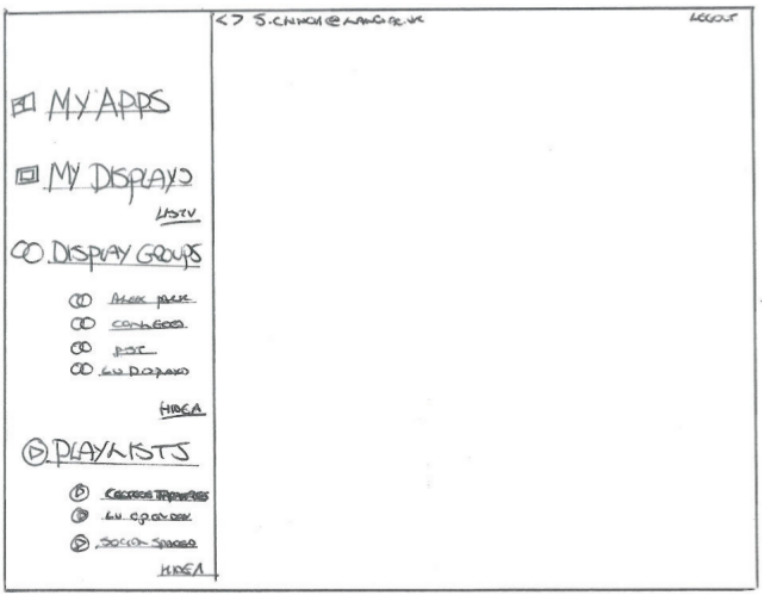


Figure 4.10: Application store for pervasive displays: web user interface outline.

Figures 4.11 to 4.13 show some sample workflows generated as part of our design process. Figures 4.11 and 4.12 show actions completed in a display owner role (browsing and purchasing applications, browsing display hardware) and Figure 4.13 shows the process of adding new content to the store which is completed by an application developer.

4.4.2.2 RESTful APIs

We decided to implement the backend of our application store as a series of RESTful APIs. These APIs would provide functionality for the described web user interface but could also act as interface points for other systems. Selection of the REST design model was based on its dominance for web APIs, its scalability and suitability for creating generalised interfaces.

4.4.3 Comparison with Prior Work

The emergence of displays that can execute multiple applications is relatively new, and selection and distribution mechanisms for such applications are therefore limited. The e-Campus platform [SFD06a], whilst restricted to traditional con-

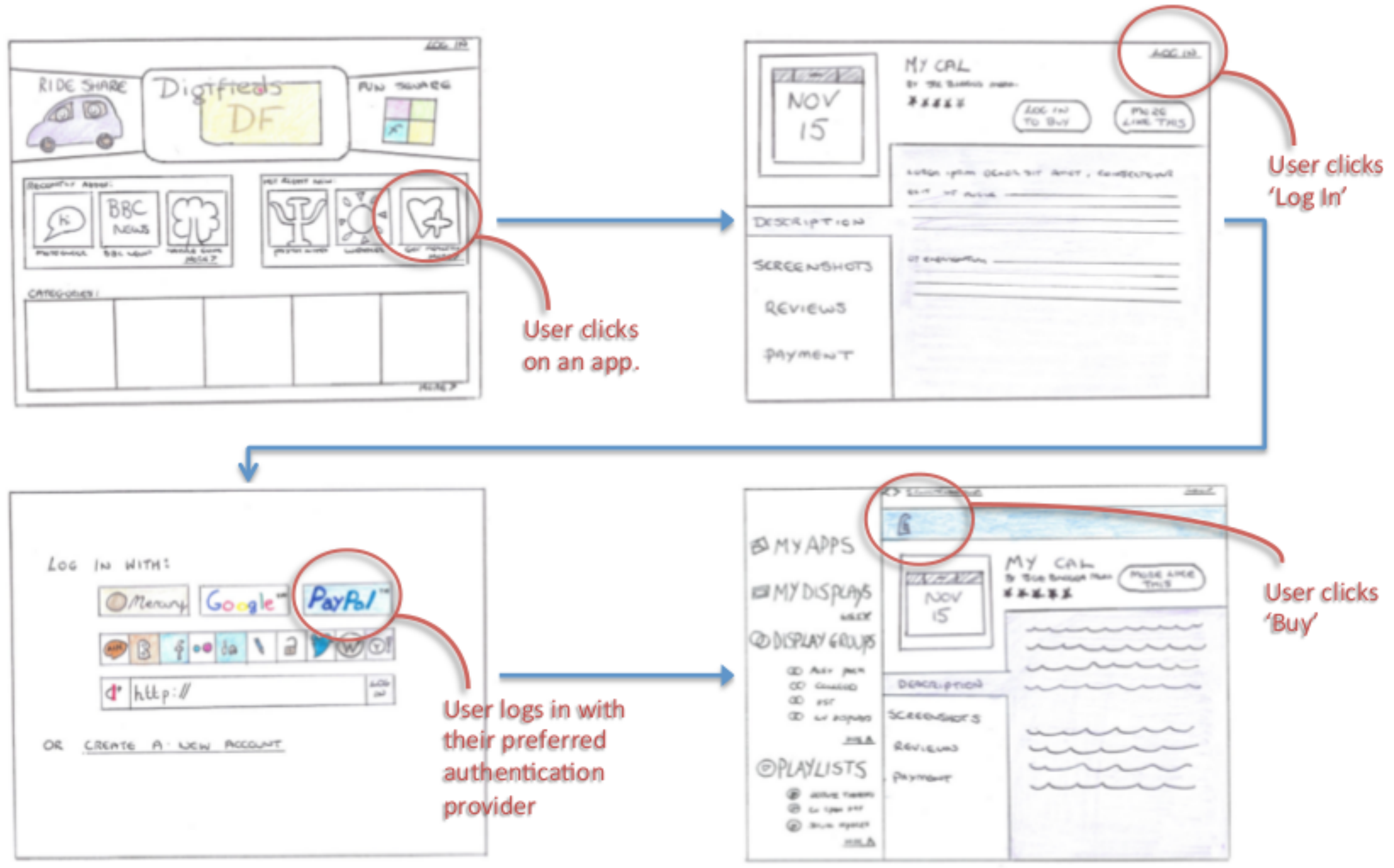


Figure 4.11: Sample Mercury web interface workflow. This workflow shows how a user can browse the application store, view application details, log in, and purchase an application.

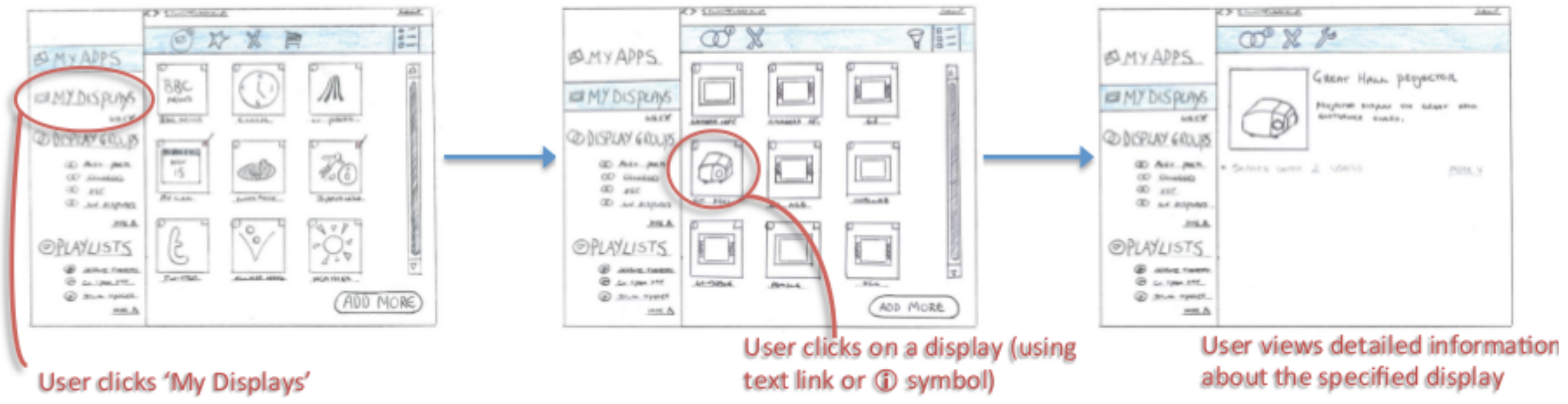

MY APPS
MY DISPLAYS
DISPLAY GROUPS
PLAYLISTS
BBC
ADD MORE
Great Hall projector
User clicks 'My Displays'
User clicks on a display (using text link or ⓘ symbol)
User views detailed information about the specified display

Figure 4.12: Sample Mercury web interface workflow. This workflow shows how a display owner can browse the display hardware that they have registered with the application Store.

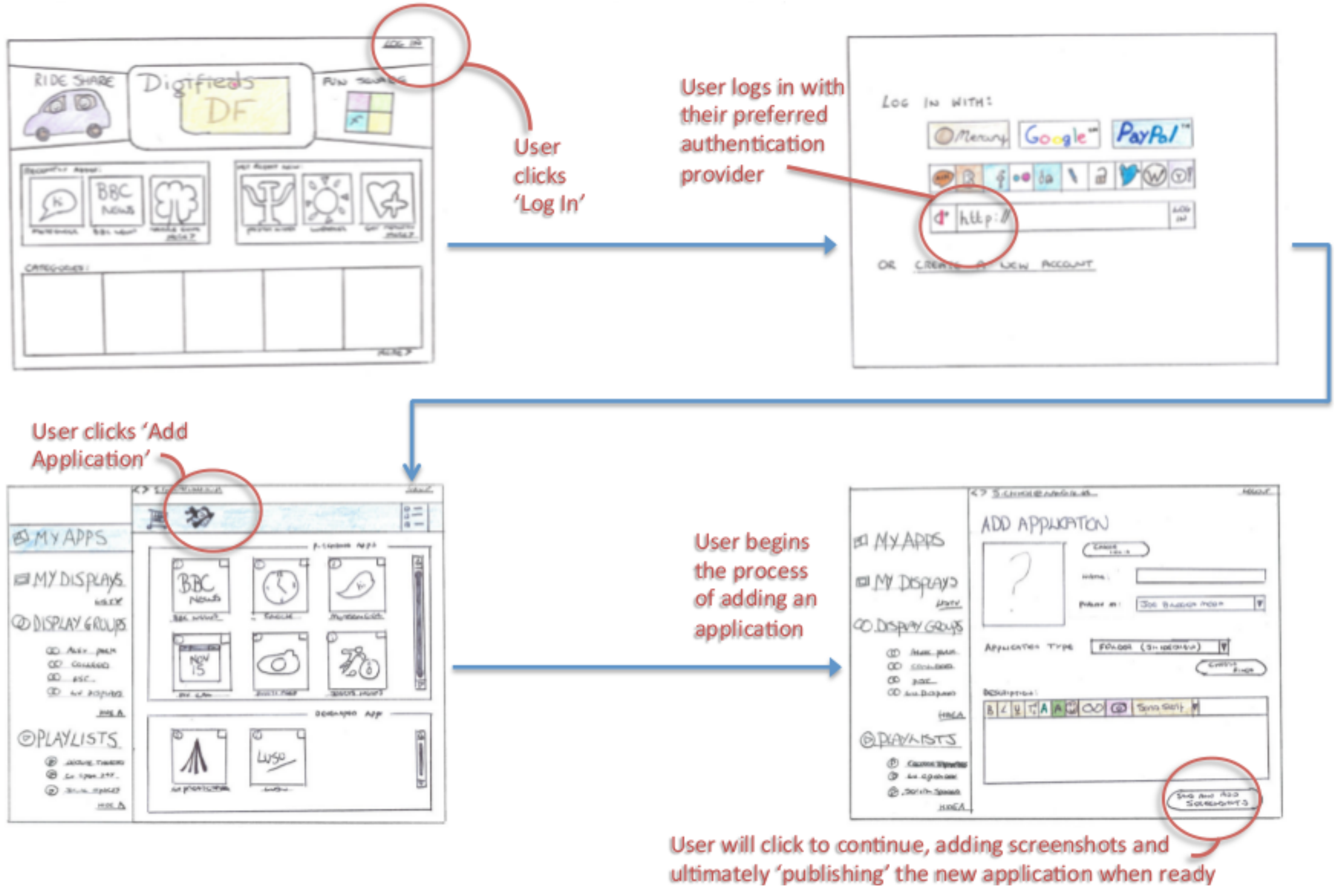

User clicks 'Log In'
User logs in with their preferred authentication provider
User clicks 'Add Application'
User begins the process of adding an application
User will click to continue, adding screenshots and ultimately 'publishing' the new application when ready
RIDE SHARE
Digifieds
DF
LOG IN WITH:
PayPal
Google
OR CREATE A NEW ACCOUNT
MY APPS
MY DISPLAYS
DISPLAY GROUPS
PLAYLISTS
ADD APPLICATION

Figure 4.13: Sample Mercury web interface workflow. This workflow shows how an application developer (content provider) can add a new application to the application store.

tent, provided a similar separation between display owners and content providers through e-Channels [CDFE11a, FDE12]. Content items could be placed into network file shares representing 'channels'; display owners then subscribed their displays to one or more channels and the associated content was then added to a database of available media items for that display.

Both the InstantPlaces [PaNR13] and UBI-hotspot [OKK+12] deployments provide support for a selection of applications to be played at displays. The InstantPlaces *application registry* [PaNR13] serves a similar role to Mercury: display owners can register their displays with the registry and install applications on them. However, creation and manipulation of schedules for the displays is managed by a separate *orchestration service*. By contrast, on the UBI-hotspot displays application selection is offloaded to the displays themselves [OKK+12] – a series of menus make multiple applications available on the screen simultaneously, allowing display viewers rather than display owners the opportunity to choose between the applications for a display.

Mercury takes inspiration from the success of application stores in the mobile domain. An exploration of the differences between Mercury and existing mobile phone application stores has been presented in Section 4.4.1 and is not repeated here.

4.5 Tacita: Supporting Personalisation/Appropriation Requests

Finally, we require support for viewer submission of display appropriation requests. Based on our requirement R3, we particularly focus on providing a mechanism for viewers to use their personal mobile devices to submit display appropriation requests. Our design focuses on Network Segment components (see Figure 4.1) and is intended to address requirements R3, R4 and R5.

4.5.1 Design Considerations

4.5.1.1 Supporting Multiple Usage and Presence Models

Chapter 3 described our three viewer appropriation usage models: walk-by appropriation, active appropriation, and longitudinal appropriation. Personalisation in existing display systems is typically limited. Research prototypes have tended towards small-scale deployments and each has focussed on a single usage model (most frequently walk-by or active appropriation). In this thesis, we aimed to design a system with support for all three appropriation usage models.

Walk-by, active, and longitudinal appropriation each have different requirements in terms of user detection and positional accuracy; whilst it is highly important that an 'active' user is co-located closely with the display, for longitudinal appropriation a user may be located simply within the same shopping precinct or city neighbourhood. Supporting these varied requirements is best achieved through a range of location technologies and presence models.

In prior research detecting viewers at a display has typically relied on short-range communication technologies such as NFC, infrared and Bluetooth (for example, FLUMP [FWDF96] and Instant Places [JOIH08][1]). In these approaches one of either the display or a viewer's personal device broadcasts a signal to be detected by the other; in this way the two devices can be determined to be co-located in space. Within the mobile device domain, GPS and Wifi positioning dominate as location technologies. Using these approaches of locating a mobile device, display proximity can be calculated by comparing the mobile device location with the known locations of a set of displays.

Used independently, NFC, Bluetooth, GPS and Wifi positioning each have strengths and weaknesses as methods for detecting viewer proximity with pervasive displays. We therefore chose to support multiple methods of viewer detection (that could be used either alone or in combination) in order to allow more accurate viewer detection in a wide range of situations and to provide good support for our three distinct usage models. Furthermore, the selection of both viewer location (e.g. GPS, Wifi positioning) and proximity (e.g. NFC, Bluetooth) ensures a degree of openness in the design allowing easy integration of additional

[1]These systems are described in more detail in Section 2.5, pages 82 and 88.

methods as needed.

4.5.1.2 Viewer Privacy

Personalised pervasive displays require the disclosure of viewer preferences and/or interests. Chapter 3 highlighted two groups of privacy concerns resulting from this disclosure: disclosure through the display of inappropriate or revealing content and disclosure through the construction of tracks and profiles.

A number of systems have attempted to address the problem of disclosure through the display of inappropriate or revealing content. For example, Section 2.2.7.2 described a variety of research prototypes that used personal mobile devices as private co-displays. Furthermore, our own studies of personalisation through Bluetooth device names (as presented in Section 3.3.1) indicate that viewers themselves may have awareness of this concern, as demonstrated by their unwillingness to use the system for personally revealing social networking content.

Our second concern, disclosure through the construction of tracks and profiles is perhaps more difficult to address. In order to support personalisation, viewers must reveal their personal data. In a typical approach this viewer data is shared with the display infrastructure either as individual pieces of information shared on a per-request basis or in the form of a more descriptive profile that can be accessed on demand. Assuming the current situation of many hundreds of thousands of displays owned and maintained by different organisations, achieving personalisation through this approach results in data sharing with many different display providers.

Examining current digital behaviours on personal mobile devices and the World Wide web, we see a trend for users to develop trust relationships with applications and service providers (e.g. Facebook, Google, Instagram). If we apply a similar model to display personalisation, we can envisage a scenario in which in order to personalise a display viewers share information with their preferred application providers rather than with the display infrastructure itself. This approach could allow viewers to share data with a known set of trusted application providers rather than an unknown and changing infrastructure.

4.5.1.3 Supporting Multiple Concurrent Personalisation Requests

We have previously described how scheduling techniques can be used to allocate the available screen space to different content items. In order to allow multiple viewers' personalised content to be shown responsively, space multiplexing can allow many items to be shown simultaneously. However, space multiplexing has clear limitations in terms of the number of concurrent content items that can be supported whilst still maintaining readability.

As an alternative (or complimentary) approach, display applications themselves can offer mechanisms for supporting multiple concurrent users. Aggregating multiple viewer requests within a specific application can reduce the number of independent application instances to be shown at a display and therefore increase the available screen space. For example, given a scenario in which three viewers are requesting sports news (two for football, one for basketball) and two are requesting local weather information, the screen can be divided into two sections rather than five, with the sports news application creating a feed containing a mixture of both football and basketball content.

4.5.1.4 Signage System Integration

Early in this thesis we recognised the very large deployed base of digital signage systems and pervasive displays. Future display networks will not replace these deployments but will integrate existing hardware. For this reason, our architecture for appropriation must accommodate existing deployments and have a low barrier to adoption for the creator of new signage systems.

We have already identified two classes of technologies for determining viewer proximity to digital displays. In location methods (e.g. GPS) the placement of deployed displays must be known in order to compare viewer position with display location. By contrast, in proximity methods (e.g. NFC) the display must be fitted with additional hardware in order that a signal can be broadcast and/or received.

By supporting multiple methods of determining viewer proximity to digital displays we can keep demands on existing systems to a minimum. Using a viewer's personal mobile device to identify their location can allow proximity detection for displays with no short-range communication hardware whilst use of proximity-based methods (e.g. Bluetooth) can remove the need to record the location of a large number of already deployed displays.

4.5.2 System Design

Based on these design considerations, we selected an existing design concept to integrate with our architecture in order to provide support for viewer submission of display appropriation requests. The design for this system, Tacita, came from an initial concept created by Nigel Davies and Marc Langheinrich and was refined by a collective that included the author of this thesis.

Tacita allows viewers to use their personal mobile devices to submit display appropriation requests. In order to do this, the design is centred around an application that executes on a viewer's personal mobile device. This mobile application monitors a viewer's changing circumstances in order to identify when they enter into proximity with an display that supports personalisation with content known to be interesting to the viewer. The mobile application then makes a personalisation request on behalf of that viewer. Once the viewer moves out of the viewing area for the display, the mobile application stops requesting personalisation.

Tacita provides the viewer with two mechanisms for the discovery of displays in their environment:

1. Each display can announce their proximity and capabilities through short-range communication technologies (e.g. Bluetooth) or visualisations at the display itself (e.g. a QR-code overlay).

2. Each display can register its proximity and capabilities with a map-provider service which can then distribute this information to mobile devices.

Both mechanisms rely on the distribution of *capability announcements* from the display to the viewer – this is in contrast to existing personalisation prototypes (e.g. FLUMP [FWDF96], InstantPlaces [JOIH08], C4 [MCH08]) which rely on viewers announcing their presence to the displays around them.

4.5.2.1 Capability Announcements

The capability announcements provide the Tacita mobile application with all the information needed to determine if a display would be a good target for a viewer's personalised content. Capability announcements must contain a display URI, a description of the geographic scope of the display, and a list of supported applications (and their associated parameters); additional data relevant to the display may also be included.

Display URI
The display URI provides a unique identifier by which the viewer can describe a display (e.g. to issue a personalisation request for that display).

Geographic Scope
We deliberately decouple the statement of geographic scope from any specific networking or location technology.

In our design considerations we highlighted the limitations of both the short-range wireless communications typically used in display personalisation prototypes, and the location technologies used in mobile applications. The design therefore incorporates a mechanism for describing geographic scope in a more abstract manner. We allow the definition of arbitrary trigger zones around a display that may be described in terms of true geographic location and/or RF propagation. The flexibility of our trigger zone definition also allows the scope to be optimised for the 3D viewing range of each specific display deployment.

In order to preserve viewer privacy, the broadcasting of geographic scope also allows a viewer to discover whether they are in range of a display without revealing their location to the physical infrastructure. The Tacita mobile device can combine available location technologies with geographic information from the capability announcement; with simple processing it can quickly be established whether the mobile device is contained within the geographic scope of the display.

Applications and Parameters
Capability announcements include a list of supported applications (in the form of URIs). Each application represents an entity with which a viewer may have already formed a trust relationship. Taking advantage of these trust relationships allows viewers to appropriate a display without having to reveal personal data directly to the display.

To allow personalisation of in-application content, each application can list a set of customisable parameters; this parameter set is broadcast as part of the capability announcement. For example, a news application might take a series of interest keywords that can be used to ensure that the stories shown have greater relevance to the viewer.

Additional Data
Finally, capability announcements can also support a small amount of additional data about the display. This can be used by the display for a wide range of purposes and may or may not be processed by client software. Sample data may include details of the currently playing item at a display, the hardware configuration, or a presence token that allows a user to prove colocation with the display.

While decoupling capability announcements from physical proximity offers significant flexibility, there are situations in which display personalisation should only be offered when a viewer is truly collocated with a display. Issuing a presence token over short-range communication as part of (or separately to) the capability announcements provides a mechanism for supporting this guarantee.

A presence token takes the form of a temporary numeric identifier, typically a pseudo-random number. The token can be updated periodically to avoid caching or sharing of tokens when away from the display. To prove proximity to the display a viewer must include the relevant presence token with its personalisation request.

4.5.2.2 Personalisation Requests

Figure 4.14 shows the primary workflow through Tacita. In this workflow a viewer uses the Tacita mobile client to fetch capability announcements either directly

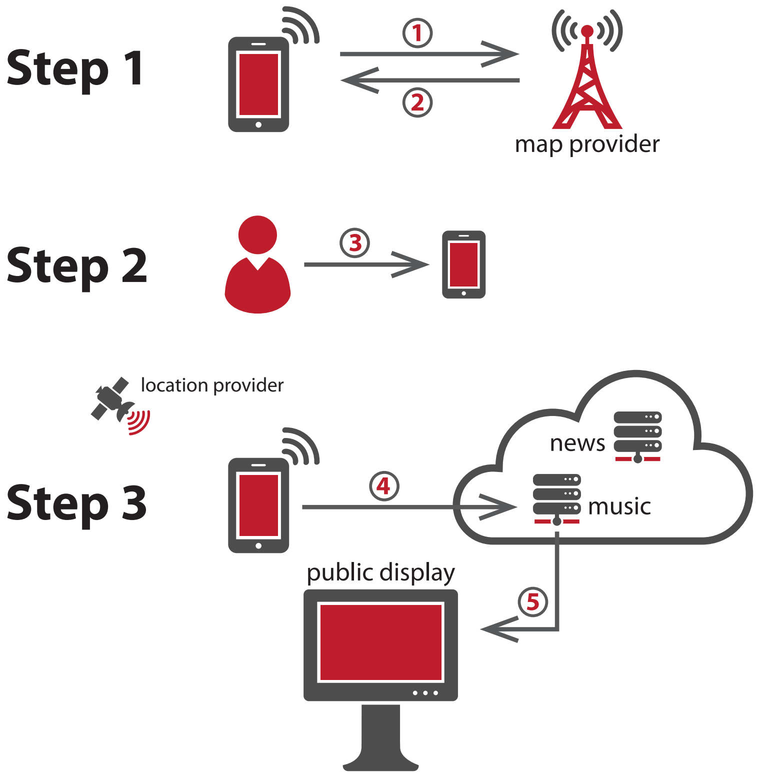


Figure 4.14: Tacita: Display personalisation workflow (previously published in Davies et al. [DLC+14]). In Step 1, the mobile client obtains a display announcement from a map provider or broadcast directly from a nearby display. In Step 2, the viewer selects a set of applications that they wish to use for personalisation. In Step 3, the entry of the mobile client into a display's geographic scope triggers a request to the viewer's cloud-based applications resulting in a scheduling request from the cloud- application to the display.

from nearby displays or as *display maps*, collections of capability announcements groups together by location and shipped to displays on request by a Tacita map provider. Based on the set of all available applications, a viewer may select a subset of those interesting to them and customise them using the associated parameters.

Whilst Tacita allows viewers to discover nearby displays and their capabilities without revealing their presence to the display infrastructure, it is clear that once a viewer decides to make a personalisation request some information must be shared to generate the resulting content. In contrast to existing display personalisation systems, Tacita does not require viewers to share this data with the infrastructure, but instead takes advantage of existing trust relationships between the viewer and application providers [Figure 4.15].

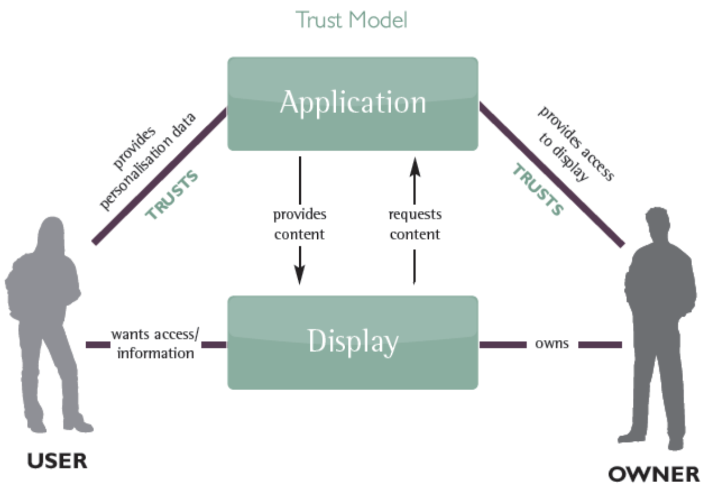


Figure 4.15: Tacita: Overview and trust relationships (previously published in Davies et al. [DLJS12]).

In order to make a personalisation request (Figure 4.14, Step 3), viewers indicate their desire for content to be customised or for space on a display to be appropriated directly to the (typically cloud-based) applications that produce the content for the display concerned and not to the display itself. Each application request should include application-specific customisation parameters plus the URI of the display at which the viewer would like to see that application and any associated presence tokens. Upon receipt of a request, the application can generate the resulting content and issue a request to the specified display to schedule the generated content item (including a presence token if supplied). While the viewer remains in the display's trigger zone, the Tacita mobile client sends keep-alive messages to the applications. Once the viewer moves away from the display and leaves the trigger zone, the requests time out and the personalised content is removed from the display.

4.5.2.3 Supporting Multiple Simultaneous Viewer Requests

Our design considerations highlighted how sending personalisation requests can introduce opportunities for aggregation of viewer requests, reducing the number of personalised content items to be scheduled concurrently at a display. The design actively encourages these kinds of application optimisations. This allows a display to potentially handle a large number of concurrent viewer requests.

Application optimisations may also offer other benefits. For example, some services may offer content anonymisation or obfuscation services to help reduce the risk of disclosure through the display of inappropriate or revealing content (one of our two main privacy concerns). For example, an application may detect and obscure personal information within a content item, allowing a viewer to retrieve the missing details through another mechanism (as in previous systems such as those by Berger et al. [BKN05b] and Sharp et al. [SSB06][1]).

4.5.3 Comparison with Prior Work

Section 2.5 described a wide variety of platforms for allowing viewers mechanisms for personalising or appropriating digital displays. At one extreme, remote graphics (e.g. RDP [Mic14], VNC [ATT99]) and cyber-foraging systems (e.g. Kimberley [WHCS08]) can allow a viewer complete control over a display by connecting back to a desktop of applications prepared ahead of time by the controlling viewer. At the opposite extreme, many prototypes restrict viewers to customisation of a restricted set of a parameters or applications (as in InstantPlaces' Bluetooth personalisation [JOIH08]). Our selection of Tacita for integration into

[1]These systems are described in more detail in Section 2.2.7.2, page 55.

the overall architecture allows support for any application that accepts customisation parameters, making it a good match for this second type of personalisation. However the architecture is also open to more flexible appropriation, and our use of Tacita also provides a mechanism for these personalisation requests.

Addressing privacy concerns when personalising digital signage has also been a focus of prior work. Like many personalisation systems (e.g. [JOIH08, MKHS08]), Strohbach et al. [SKM09b] used Bluetooth devices to personally identify viewers in order to determine how to personalise their displays. However, Strohbach et al. shared Tacita's interest in preserving viewer privacy by avoiding broadcast communications. Strohbach et al. addressed privacy concerns by requiring viewers to register their device's Bluetooth MAC address – known MAC addressed could be searched for in a manner that still allowed viewers to keep their phones out of discoverable mode and so avoided observation by third-parties. Strohbach et al.'s architecture made no efforts to avoid tracking by the display infrastructure itself, indeed identification of viewers by the displays was required in order to personalise content. By contrast, Tacita explicitly protects viewers from having to reveal their devices and preferences from not only third-parties but also the displays themselves.

Whilst digital signage personalisation techniques have typically focussed on proximity detection through RF propagation techniques (e.g. having displays receive data sent over Bluetooth [JOIH08, MKHS08] or infrared [FWDF96]), Tacita's requirement to avoid revealing personal data to display infrastructure has resulted in proximity detection occurring at the viewer's personal device rather than the display. The Tacita mobile client combines multiple location technologies to detect a viewers position and therefore proximity to displays. Similar location-based models of smart environments have also been seen in prior work (e.g. [ACH$^+$01]).

Tacita's proximity detection relies on the display infrastructure itself broadcasting out capability announcements that describe the applications available for personalisation. The idea of using infrastructure-based broadcasts to improve user privacy is not new – Langheinrich's pawS [Lan02] required smart environments to advertise their capabilities in order to allow potential viewers to identify services and their associated privacy agreements. In pawS, data exchange with acceptable services then takes place through a privacy proxy, in Tacita personalisation is proxied through trusted applications that pass back the customised content to be rendered.

Tacita was conceived by Nigel Davies and Marc Langheinrich; the mobile clients were developed by Thomas Kubitza and Christopher Winstanley. While related work has explored display personalisation, our use of Tacita as part of an overall architecture to support viewer appropriation of open pervasive display networks is the first such initiative.

4.6 Summary

This chapter has detailed the design of an architecture for appropriation of pervasive display networks comprised of three key components. Yarely, our client scheduler and media player is designed to support both day-to-day digital signage needs and a wide range of future content types. Our second component, Mercury, acts as an application store for public display applications allowing easy distribution of applications to many displays. Content schedules issued by our Mercury component and other content descriptor factories take the form of a Content Descriptor Set (CDS). Finally, Tacita provides a mobile client and map provision service for supporting appropriation requests whilst withholding a user's identifying features from the display hardware itself.

In the next chapter, we will describe the implementation of each of our three key components and the content descriptor set format used to exchange content schedules.

Chapter 5

Implementation

5.1 Overview

The previous chapter outlined the design of an architecture for appropriation of pervasive display networks. In this chapter we describe the implementation and deployment of this architecture.

We begin by identifying a set of core scheduling and playback components that form the implementation of Yarely and describe the resulting system developed for Mac OS X and Linux. We then specify a realisation of our content descriptor sets through an XML document format. We describe the development of a system for production of content descriptor sets, that of the application store (Mercury). Next we provide implementation detail for the final portion of the architecture, that of Tacita, used to support the user in the actual process of making an appropriation request. Finally, we describe the physical and geographical characteristics of our deployment.

Work in this chapter has also been published in:

- Thomas Kubitza, Sarah Clinch, Nigel Davies, and Marc Langheinrich. Using mobile devices to personalize pervasive displays. In *Demo. at HotMobile '12*, 2012 [KCDL12].

- Sarah Clinch, Thomas Kubitza, Nigel Davies, and Marc Langheinrich. Demo: Using mobile devices to personalize pervasive displays. In *Demo. at Mobisys '12*, 2012 [CKDL12a].

- Sarah Clinch, Thomas Kubitza, Nigel Davies, and Marc Langheinrich. Using mobile devices to personalize pervasive displays. In *Demo. at Digital Futures '12*, 2012 [CKDL12b].

- Sarah Clinch, Nigel Davies, Adrian Friday, and Graham Clinch. Yarely – a software player for open pervasive display networks. In *Proceedings of the 2013 International Symposium on Pervasive Displays*, PerDis '13, New York, NY, USA, 2013. ACM [CDFC13].

- Nigel Davies, Marc Langheinrich, Sarah Clinch, Adrian Friday, Ivan Elhart, Thomas Kubitza, and Bholanathsingh Surajbali. Personalisation and privacy in future pervasive display networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, New York, NY, USA, 2014. ACM [DLC+14].

A summary of this author's contributions to the above publications and the work presented in this chapter is given in Table 4.1 (page 150).

5.2 Yarely: Client-Side Scheduling and Media Playback

5.2.1 Platform and Tools

The Yarely software player is implemented in Python 3 [Pyt13e] and we use the ØMQ transport layer [iMa13] to transmit events between portions of the software (fulfilling the role of the internal eventing system shown in Figure 4.3). PyZMQ [GRK13] provides the required Python bindings for communicating over ØMQ. Data exchanged over ØMQ is encapsulated using XML.

The use of Python as the primary programming language was motivated by a requirement to produce software to control a heterogenous hardware set. Python is a cross-platform programming language, with CPython currently providing support for a range of Unix-like platforms (including Mac OS X), Microsoft Windows 2000/NT–8, and gaming and mobile platforms (Nintendo NS, Xbox, Android etc.) amongst others. The ØMQ library provides support for over forty programming languages and popular operating systems, again helping to meet our requirement for a cross-platform software.

On the Mac OS platform, the Cocoa API is the native operating system API. Cocoa consists of three Objective-C object libraries: Foundation Kit, Application Kit, and Core Data. We use PyObjC [OBM+13] to bridge to Objective-C, allowing us to use and extend the Cocoa objects within our Python code. On Linux, we use the GNOME libraries GObject, GLib and GTK and use PyGObject [GP11] to bridge to GObject-based libraries such as `WebKitGTK+` [WT13] and `gtk-vnc` [GP13].

A small set of sensor handlers are developed to handle incoming data over cross- platform network technologies. The HTTP server sensor handler uses the Tornado web framework [Fac13] to handle incoming HTTP client requests.

Use of extreme programming coding and testing techniques can help to ensure software quality. Throughout the development process, a number of tools were used to help ensure resilient, robust code. The passive Python testing tool Pyflakes [Pyt13d] was used to detect unused library imports and undefined or unused variables. The code was written to comply with the PEP 8 [Pyt13b] Python code style conventions and the `pep8` tool [Pyt13c] was used to check compliance with this standard. Active tests were written and executed using the `unittest` [Pyt13f] and `doctest` [Pyt13h] modules.

Code was versioned using the Kiln [Fog13a] (Mercurial [Mac13]) version control system and repository tags determined whether a commit was pushed out to live deployments, a small testbed or simply remained in the repository. In addition, a Buildbot [Bui13] continuous integration system connected to web hooks allowed us to trigger our passive and static tests following each commit to the code repositories; all tests for cross-platform code were executed on multiple platforms (Windows XP, Windows 7, Mac OS 10.6, Mac OS 10.7, Ubuntu 11.10) and platform specific code was tested on all relevant platforms. Development tasks and a code review process were managed using Fogbugz [Fog13b].

5.2.2 Implementation

Figure 4.3 (page 158) shows our original architectural design consisting of five components: Subscription Management, Sensor Management, Playlist Generation and Scheduling, Content Rendering and Lifecycle Management, and Analytics. These components are represented in a series of Python packages – our first three components (Subscription Management, Sensor Management, Playlist Generation and Scheduling) are implemented predominantly as cross-platform code, whilst the Content Rendering and Lifecycle Management components is implemented once for each of our target platforms. Our current implementation does not include an Analytics module.

Subprocess Management

The architectural design (Figure 4.3) featured a repeated pattern of components comprised of one Manager plus a series of Handlers. This pattern is implemented as a Python process for each Manager element with a series of subprocesses for the associated Handlers. Communication between the Manager and its Handlers is achieved through ØMQ Request-Reply socket pairs. Subprocess management is provided once as a set of abstract base classes implemented in the `core` package.

Package Layout

As shown in Figure 5.1 Our Yarely implementation is divided into four subpackages: `core`, `darwin`, `linux`, and `windows`. The `core` subpackage represents primarily cross-platform code whilst the remaining three subpackages are specific to the target platform; each deployment of Yarely therefore requires two subpackages, `core` plus one of the platform packages.

Our implementation focuses on support for Macintosh (targeting Mac OS X 10.6 and above using the `darwin` subpackage) and Linux with GNOME libraries (using the `linux` subpackage). An adaptation of the `linux` subpackage was also produced to target the Raspbian (Raspberry Pi) platform. We include a `windows` subpackage as a placeholder for future work.

Table 5.1 shows the distribution of code between the `core` and platform-specific sub packages.

	Python Files	Lines of Code
Cross-Platform	**58**	**7587**
Subscription Management	9	1344
Sensor Management	4	413
Scheduling	6	1802
Configuration	3	501
Unit Tests	8	388
Other (helper modules, base classes etc.)	28	3139
Mac OS X	**24**	**1415**
Rendering	8	411
Facade	3	306
Other (helper modules, base classes etc.)	13	698
Linux (GNOME)	**11**	**166**
Rendering	4	33
Other (helper modules, base classes etc.)	7	133
Linux (Raspbian)	**8**	**482**
Scheduling	1	55
Rendering	2	180
Other (helper modules, base classes etc.)	5	247

Table 5.1: Proportion of cross-platform versus platform-specific code within the Yarely subpackages.

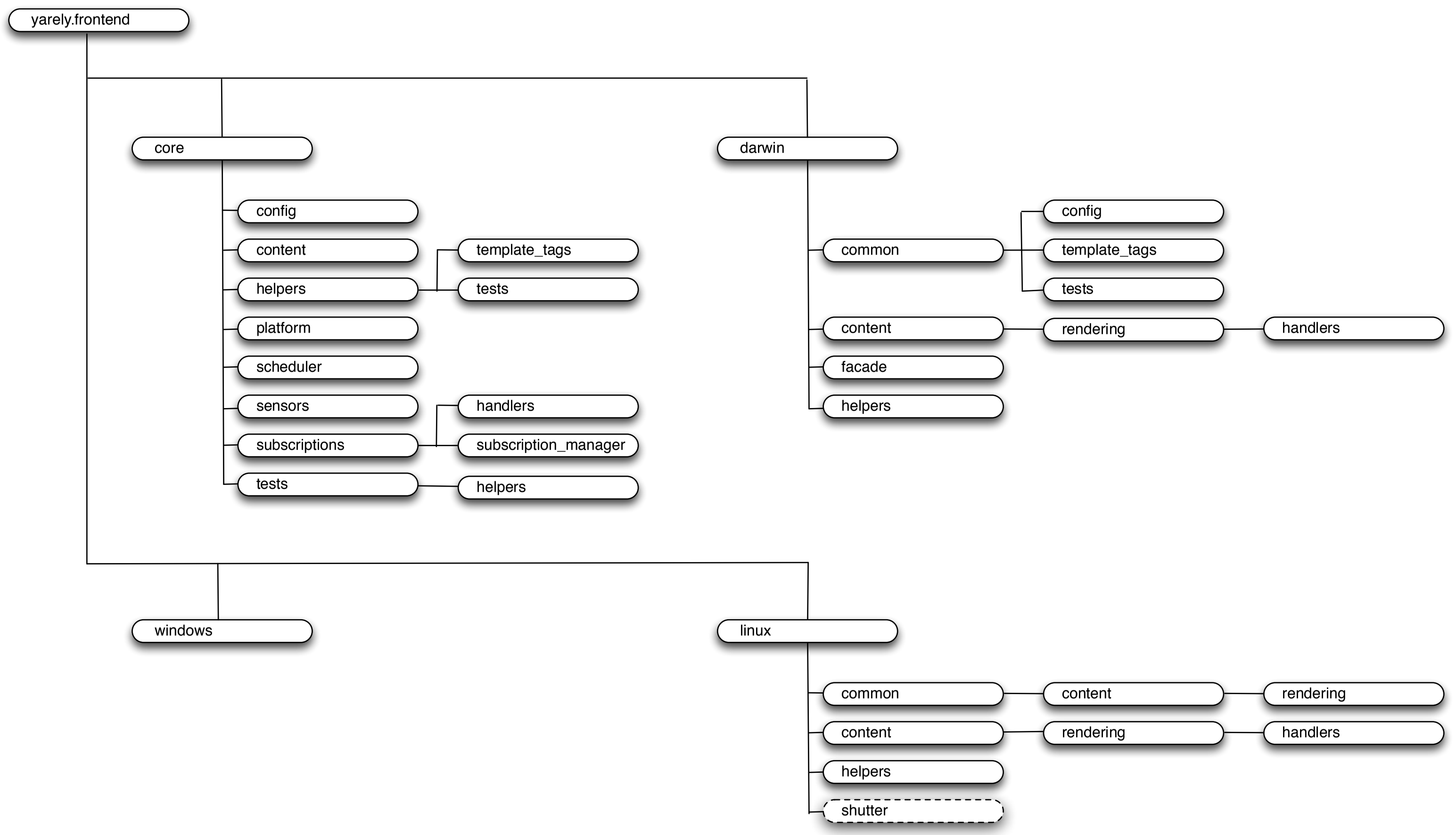


Figure 5.1: Python package and subpackage hierarchy for the Yarely implementation. The dotted `shutter` subpackage is specific to the Raspberry Pi (Raspbian) version of the `linux` subpackage. The `windows` subpackage is provided only as a placeholder for future work.

5.2.2.1 The `core` Subpackage

Our initial implementation maps each of the Subscription Management, Sensor Management, and Playlist Generation and Scheduling components to separate Python subpackages contained within the `core` package (`core.subscriptions`, `core.sensors`, and `core.scheduler` respectively). Lifecycle Management is divided between the `core` and platform-specific subpackages, with the majority handled in the platform-specific subpackages. Each of our four implemented modules is represented in the subpackages by a number of Python modules and one or more applications.

In the following paragraphs we provide a detailed overview of the `core` subpackages and the architectural components implemented within them.

Utility code

The `core` subpackage contains three modules that do not correspond directly to any of our five architectural components, but instead provide utility functionality used by modules in all Yarely packages.

The `core.config` subpackage provides utility functionality through a number of classes for reading and writing configuration files. Yarely config files are written in the INI file format. Yarely configuration can be used to specify directories to which Yarely can write data (e.g. file caches, application state, logging), detail display hardware information (e.g. interface addresses), and manage the overall signage experience (e.g. default duration for simple media types, background image when no content is playing).

The `core.platform` subpackage provides platform-specific utility functions such as checking disk space and generating valid file path URIs.

Finally, the `core.helpers` subpackage provides base classes, decorators, execution control loops, data types and conversion libraries, custom loggers, and helper libraries for communicating over ØMQ and RFB (for VNC).

Together the modules in `helpers`, `config`, and `platform` abstract over common functionality for applications within the `core` subpackage such as reading configuration, initialising logging, managing subprocesses, and responding to signals.

Scheduling and Lifecycle
The `core.content` and `core.scheduler` subpackages contain modules that correspond to the *Content Rendering and Lifecycle Management*, and *Playlist Generation and Scheduling* portions of the architectural design.

Content Rendering and Lifecycle Management is predominantly a platform-specific operation and will therefore be discussed further in Section 5.2.2.2. Within the `core` module however, support is provided for caching content files prior to playback. Each file is cached immediately prior to the first time it is due to appear at the display. If the file cannot be cached at that time then the item is temporarily removed from the playlist. Items are restored to the playlist (resulting in a new caching attempt) in response to the next subscription update. The content cache is cleared each time the display node is restarted.

The Playlist Generation and Scheduling component is provided by the `scheduler.py` application in the `core.scheduler` subpackage. Playlist generation is triggered by receipt of a `subscription_update` XML message over the ØMQ channel; this message is generated by the `subscription_manager.py` application in the `core.subscriptions` subpackage. In order to generate a playlist, we examine the constraints of each media item contained within the current subscription set (including inherited constraints) and remove all items that cannot be played at this time.

Our initial scheduler cycled through each of the available media items using a round-robin algorithm. Prior to playback, each item would first have its constraints checked to see if the item was valid for the current circumstances (based on time, date and day-of-week constraints specified in the Content Descriptor Set – these will be discussed further in Section 5.3). The scheduler then generated a playlist based on the current set of valid items, and used this playlist to generate a local file cache. As the scheduler prepared to play the next item in the playlist it would check the cache to ensure a local copy of the media could be found. The scheduler would attempt to play the local file for the length of time specified by the `preferred-duration` constraint or a default value if this constraint was not specified.

Our current scheduler extends this functionality to add priority and ratio support (as described in the Content Descriptor Set). Yarely supports five levels of priority which may be represented in the Content Descriptor Set either numerically, as a value from zero (lowest) to four (highest), or as a text string with one of the following values: lowest, low, medium, high, highest. The addition of this set of priority levels allows us to responsive content in a range of situations including emergency announcements and personalised content. Our support for ratios allows groups of content items to be assigned a proportion of the available airtime and is implemented through using a credit-token-based rate pacing scheme. This ratio functionality provides finer-grained control over scheduling and also ensures that content items with a long-duration do not dominate screen-time when compared to shorter-length items.

In order to provide support for viewer appropriation, the Yarely scheduler can alter the current playlist in response to incoming sensor updates (received over the ØMQ event channel). Upon receipt of such a request, the scheduler can break out of the current content schedule and instead schedule the requested items. Once receipt of sensor updates stops, the dynamically scheduled content is removed and the regular content schedule is resumed. Our scheduler provides support for space-multiplexing of viewer-driven content for up to twenty concurrent users. In addition to supporting viewer appropriation of the display, this dynamic schedule update mechanism in response to sensor data could also be used to provide context-triggered content (e.g. showing particular content items in response to a change in the noise or light levels in the environment around the display).

A final function of the `core.scheduler` subpackage is power management of the physical display device(s). Control of the display hardware is achieved through an additional application, `display_controller.py`. Communication between the scheduler and display controller processes takes place over the ØMQ event channel. When a content item is ready to be scheduled at a display, the scheduler sends an event to the display controller requesting that the display is turned (or remains) on until a short time after the content item is expected to be removed from the display.

Sensing
The Sensor Management component is provided by the `sensor_manager.py` application in the `core.sensors` subpackage. The sensor manager creates a new handler for each socket to be managed and upon receipt of data from that socket, forwards the data to all interested parties through the ØMQ event channel. Forwarded data is enclosed inside a `sensor_update` XML element.

Our initial implementation provides a socket sensor handler which accepts incoming data over a TCP connection and a HTTP server sensor handler which accepts incoming data from HTTP clients.

Subscriptions
The Subscription Management component is provided by the `subscription_manager.py` application in the `core.subscriptions.subscription_manager` subpackage.

We use the Content Descriptor Set (CDS) format for describing Yarely subscriptions; this format will be described in detail in Section 5.3. Each Yarely node has a single root CDS stored in a local file whose path is defined in the configuration. The subscription manager starts a handler to parse this file. For each link to a remote `content-set` a new handler will be started.

Our implementation provides two Handlers for reading subscriptions: one handler to pull from local file URIs and one to pull from URIs that use an HTTP scheme. Both handlers periodically poll the source to check for changes to the CDS. As each handler fetches a CDS, it notifies the manager over ØMQ. The subscription manager collates all updates to expand the original root CDS, replacing all remote elements with the file contents fetched from the remote source. Each update to a CDS causes the subscription manager to forward the updated CDS to all interested parties through the ØMQ event channel. Forwarded data is enclosed inside a `subscription_update` XML element.

Testing
Finally, the `core.tests` subpackage contains modules for testing modules within `core` and its subpackages. Within `yarely.core`, many operations contain `doctest`-compatible docstrings. Further to these, the `core.tests` subpack-

age contains additional tests designed to systematically check the functionality of operations provided by `yarely.core` and its subpackages.

5.2.2.2 Platform-Specific Subpackages

We provide support for three distinct platforms: Mac OS X (versions 10.6+), Linux GNOME and Linux Raspbian. The Mac OS X implementation supports rendering of images, videos, web content, stream content, and remote desktops (using VNC). The Mac OS X sub packages also include Facade, a simple no-content renderer that runs regardless of any other renderers. Facade runs behind all other Yarely windows but in front of any other applications, thereby masking the OS X desktop from passing viewers. The Linux implementations support a more restricted content set: both the GNOME and Raspbian versions support images and web content[1], our GNOME implementation provides additional support for remote desktop (VNC) rendering.

Content renderers are a specialisation of the handler functionality provided by base classes in `core`. Each content renderer executes as a separate Python application. On Mac OS X and GNOME, each content renderer manages a single content item from preparation to termination – when the content item stops being visible on the display the associated renderer is terminated. On Raspbian the limited resources of the Raspberry Pi have resulted in an optimisation in which a small pool of long-lived renderers is initially created by the scheduler. When the scheduler is ready to prepare an item to be shown, it assigns the content item to an unused renderer from the pool[2].

The platform-specific subpackages use native operating system APIs to interface with existing windowing frameworks. Our Mac OS X renderers are developed using the PyObjC bindings for Cocoa's Foundation, AppKit, QTKit and WebKit (for the default vnc, image, video and web renderers respectively) plus `libvlc` bindings for media streams (and an alternative video renderer). The Linux GNOME renderers are written using the GObject Introspection (GI) bindings together with `WebKitGTK+` and `gtk-vnc` (for the default web and VNC

[1]On the Linux platforms both images and web content are shown by the same renderer type, images are automatically scaled and centred in a simple web page.

[2]This change also accounts for scheduling code reported under Raspbian in Table 5.1

renderers respectively). Finally, the Linux Raspbian renderer is written using Uzbl [Pla13] web interface tools.

5.3 Content Descriptor Sets: Describing Content and Constraints

In the previous chapter (Section 4.3), we described how the Yarely media player could receive instructions from a set of Content Descriptor Factories that describe the content to be played. These instructions take the form of a Content Descriptor Set (CDS) and include a description of a set of content items to be played by the node, the circumstances in which they should be played, and the location of any required media.

5.3.1 Platform and Tools

We cast CDSs as documents to be exchanged over the network. We encode CDSs using Extensible Markup Language (XML). XML forms the basis of many web document formats, communication formats and data structures; numerous tools for reading and writing XML exist across a wide-range of platforms and programming languages. In addition, the format is an open, human-readable standard. We describe our CDS definition using the W3C XML Schema Definition Language (XSD) [W3C04].

Whilst we envisage transfer of CDSs as a protocol-agnostic process, our implementations focus on pull-based retrieval over HTTP. We have developed tools to write out CDSs in PHP 5 [PG13c] using the DOM extension [PG13b] and Python 2.7 [Pyt13e] using the `xml.etree.ElementTree` module [Pyt13g].

5.3.2 Implementation

We realise our design for Content Descriptor Sets (CDSs) as documents exchanged over the network between Content Descriptor Factories (CDFs) and media scheduling and playback software. Our implementation of the CDS therefore focuses on defining this document format. We declare transmission of the CDS to be protocol-agnostic and remain open to both push- and pull-based network methods for retrieval of the CDS from a CDF. We also note from our original design the many-to-many relationship between a Display Node and CDF: each Display Node may use CDSs from multiple CDFs and each CDF may serve many nodes.

Our design featured three key entities to be contained within the CDS. We represent these entities within our document as XML elements. The document's root node must be of type `content-set`. Each content set within the document may contain a `constraints` element, any number of `content-set` elements and/or any number of `content-item` elements; a `constraints` element may also occur within each `content-item` element. Listing 5.1 shows a sample CDS document.

5.3.2.1 An Overview of Key Element Types

An XML Schema (XSD) for the Content Descriptor Set document format is given in Listing A.1 (page 311). The XML elements representing the three key entities (`content-set`, `content-item` and `constraints`) are described in more detail in the following paragraphs. We also describe the `requires-file` element and its children that are used to represent links between a Content Descriptor Set element and other files (e.g. media files and additional CDS documents). All elements are designed with extensibility in mind such that a new constraint type or media file type could easily be represented without requiring alteration to existing documents (i.e. ensuring backwards compatibility).

The `content-set` Element
The root element of a CDS must be a `content-set`.

A `content-set` is essentially a container element for `content-items` and other `content-sets`. This deliberately-recursive structure of nested `content-sets` is intended to facilitate merging of content from numerous content sources.

Each `content-set` may contain a name attribute that describes the set in a meaningful, human-readable way. `content-sets` also have a `type` attribute which may contain one of two values `inline` or `remote`. If the `type` attribute is not specified, its value defaults to `remote`. An `inline` `content-set` has

```
<?xml version="1.0"?>
<content-set name="Visit Day 2012" type="inline">
  <auth handler="none"/>
  <feedback/>

  <constraints>
    <scheduling-constraints>
      <playback order="random" avoid-context-switch="false"/>
      <time><between start="09:00:00" end="17:30:00"/></time>
      <date><between start="2012-09-22" end="2012-09-22"/></date>
      <priority level="medium"/>
    </scheduling-constraints>
  </constraints>

  <content-item content-type="video/mp4; charset=binary" size="
      42026281 bytes">
    <requires-file>
      <hashes><hash type="md5">f421a52be576891e0948a07067b8dc38</
          hash></hashes>
      <sources><uri>http://content.com/vd2012-40Mb.mp4</uri></
          sources>
    </requires-file>
    <constraints>
      <scheduling-constraints>
        <preferred-duration>246.29</preferred-duration>
      </scheduling-constraints>
    </constraints>
  </content-item>

  <content-set name="7: Visit Day 2012" type="remote">
    <requires-file>
      <sources><uri refresh="2 minutes">http://content.com/channel.
          php?id=202</uri></sources>
    </requires-file>
  </content-set>
</content-set>
```
Listing 5.1: A Sample Content Descriptor Set (CDS).

its content-item and content-set elements represented directly as child elements within the XML document. A remote content-set has its child content-item and content-set elements represented in a separate CDS document; a link to the document is provided as a requires-file element as a child of the content-set element. The structure of the requires-file element is given below.

A content-set element may contain a constraints element. The restrictions contained within this constraints element will be applied to all child content-sets and content-items. The structure of the constraints element is given below.

The content-item Element
A content-item describes a single item of media that may be rendered at a display. Each content-item element must have a content-type attribute that contains a text string representation of the media type associated with this content-item. The content-type string should correspond to the internet media type identifier (MIME type [IANAI13]) for this content-item's media or application file where such an identifier exists (i.e. for most standard media formats). If a standard identifier does not already exist then the string should follow the standard format for Internet media types: "<type>/<subtype>; <optional parameters>" (e.g. "text/html; charset=UTF-8"). In addition to the content-type attribute, a content-item may also provide a size attribute containing a human-readable string representation of the total size of the data associated with this item (e.g. "42026281 bytes" or "40.0794 MiB"). Finally, like content-set elements, a content-item also has a type attribute which may be either inline or remote and which defaults to remote.

All content-items must contain a representation of the data associated with this item. The data may be represented either directly as text within the content-item element or through a set of links to other files using one or more requires-file elements. One requires-file element should be provided for each item required to render the media at a display (e.g. for most images and videos this is a single file, a web page might require an HTML file plus a CSS file, a virtual machine might require a disk image plus a memory image). The structure of the `requires-file` element is given below.

A `content-item` element may contain a `constraints` element. The restrictions contained within this `constraints` element will be applied to this `content-item` only. The structure of the `constraints` element is given below.

The `constraints` Element

A `constraints` element can be contained within either a `content-set` element or a `content-item` element. The `constraints` element is a container element for child elements that describe the circumstances in which media associated with a `content-set` or `content-item` may be shown at a display.

A `constraints` element must contain zero or one of each of the following elements which act as containers for specific types of constraints:

`scheduling-constraints` A container element for traditional signage requirements such as date, day of week, time, duration, priority, order and ratio/weighting.

`input-constraints` A container element for constraints that relate to the input hardware requirements of the display (e.g. must support gestures).

`output-constraints` A container element for constraints that relate to the output hardware requirements of the display (e.g. must have speakers).

Since `constraints` elements may be added to both `content-items` and any parent `content-sets`, it is possible for more than one set of constraints to be applied to a media item. Our implementation currently assumes that such constraints are likely to be combined with a 'logical and' operation at the display node. This combination of constraints would need to be met simultaneously if order for an item to be shown (as in our Yarely implementation).

The `requires-file` Element

A `requires-file` element can be contained within either a `content-set` element or a `content-item` element. A `requires-file` element must be present for all `content-sets` and `content-items` of type `remote`. In most circumstances only a single requires-file element is expected per content-set or content-item; for future rich display media, we anticipate that some content may require more than one file and for this reason more than one requires-file element is permitted per parent content-sets or content-item.

The requires-file element is a container element for child elements that together describe a single file, specifically a set of file hashes and sources. Each requires-file element must contain zero or one hashes elements and exactly one sources element. Multiple hash elements may be contained within the hashes element but each one must be of a different type (e.g. md5, sha1) and must describe the same file. Similarly, multiple uri elements may be contained within the sources element but each one must point to a copy of the same file.

5.3.2.2 Implementation Issues with XSD

In addition to our textual description, we use XSD to provide a formal representation of the CDS format that can be used to validate CDS documents.

Representing the format in XSD required specification of our own custom data types in addition to standard data types (e.g. string, anyURI). Our data types are shown in Figure 5.2. The content-set and content-item share a significant number of valid child elements so these are represented in a base type _withpreambleType that both content-set and content-item extend.

Use of the XSD format has a number of limitations and our XSD is therefore not a complete specification of the CDS format. Conditional structures cannot be represented by the XSD format so whilst the presence of a requires-file element becomes compulsory for all content-set and content-item elements with a type attribute of remote this cannot be specified in the schema. Instead, we set the minOccurs indicator to zero in all cases. We use an annotation to indicate this otherwise unspecified requirement for a requires-file element for content-set and content-item elements with a type attribute of remote.

In addition to the above limitation, we also note that use of the XSD format also led some elements to be more tightly-ordered than originally intended

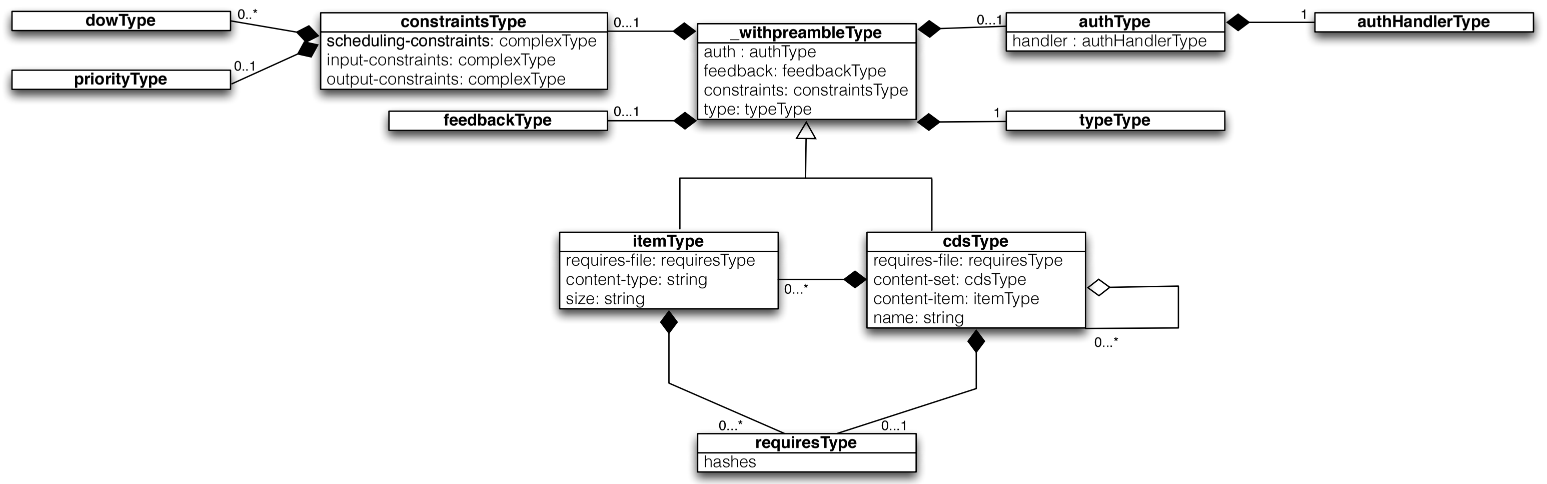


Figure 5.2: UML class diagram showing datatypes for the XSD representation of our Content Descriptor Set format. Common children of the `content-set` and `content-item` elements are represented in the `_withpreambleType` that both `content-set` and `content-item` extend. The recursive nature of `content-set` elements is shown by an aggregation relationship at the bottom right.

(e.g. an `inline content-set` must have all child `content-set` elements appearing prior to any child `content-item` elements). Finally, some of our data types are less restrictively represented in the XSD than in our textual description (e.g. Section 5.3.2.1 describes the format of the `content-type` " `<type>/<subtype>; <optional parameters>`" whilst the XSD only specifies that this value must be string.

Despite these limitations, the XSD provides a useful mechanism for allowing automatic validation of CDSs.

5.4 Mercury: An Application Store for Public Display Applications

5.4.1 Platform and Tools

Our design for Mercury incorporates two interfaces: a web user interface, and a set of RESTful APIs. The use of web applications is typical for software-as-a-service clients and REST also dominates as the primary style for web APIs. Both the RESTful APIs and the web user-interface were powered by a backend built using the Django framework (version 1.5.1) [Dja13a]. The cross-platform nature of Django (and its underlying Python (version 2.7) implementation) fitted well with our requirements. The Django framework powers a large number of websites including popular social applications such as Pinterest[1] and Instagram[2] indicating its suitability for large-scale web applications. Furthermore, Django is the dominant Python web framework with a large development community resulting in regular updates and a wide selection of plugins to provide additional functionality.

The Django framework follows a model-view-controller pattern for application design. Data models defined as Python classes (e.g. Listing 5.2) are stored in a relational database using an object-relational mapper. We selected MySQL [Ora] for the underlying database based on its dominance for web applications, and its speed, stability, and compatibility with a wide range of programming languages

[1] `http://www.pinterest.com`
[2] `http://instagram.com/`

(i.e. library support). An admin interface is also provided to allow access to the underlying data. While we did not make this interface available as part of the public UI, this did provide a useful tool for debugging. Django's view component is provided in the form of a templating language that allows template files (usually HTML) to be populated with values from the Models (e.g. Listing 5.5). Finally, the controller component uses a regular-expression based URL dispatcher (e.g. Listing 5.3) and a set of Python view handlers (e.g. Listing 5.4) to select the appropriate data from the Models and produce a response.

```python
import os.path

from django.db import models
from django.db.models.signals import post_save
from django.utils import timezone

from users.models import DevelopmentCompany
from reviews.models import Review


class Application(models.Model):
    name = models.CharField(unique=True, max_length=30)
    publication_date = models.DateTimeField(
        'date published', default=timezone.now()
    )
    icon_square = models.ImageField(
        'Icon', max_length=255, upload_to="icons", blank=True
    )
    category = models.ForeignKey('Category')
    developmentCompany = models.ForeignKey(DevelopmentCompany)

    # Description is stored in markdown
    description_markdown = models.TextField('Description')

    def get_avg_score(self):
        """Get the average rating based on user reviews."""
        reviews = Review.objects.filter(application=self.pk)
        score = reviews.aggregate(models.Avg('score'))['score__avg']
        return 0 if score is None else score
```

Listing 5.2: Mercury: Sample Django Model Snippet.

To provide additional functionality, a set of Django plugins were used. We use the Django Model Utils plugin (1.3.1) [Mey13] to optimise class inheritance

```
from django.conf.urls import patterns, url

from applications.views import AllApplicationsView
from applications.views import ApplicationDetailView

urlpatterns = patterns(
    '',

    # Listing applications
    url(r'^applications/all/$',
        AllApplicationsView.as_view(),
        name='applications_all'),
    url(r'^applications/all/page/$',
        AllApplicationsView.as_view(),
        name='applications_all_paginated'),
    url(r'^applications/all/page/(?P<page>[0-9]+)/$',
        AllApplicationsView.as_view(),
        name='applications_all_paginated'),

    # Application detail site
    url(r'^applications/app/(?P<pk>[0-9]+)/',
        ApplicationDetailView.as_view(),
        name='applications_detail'),
)
```

Listing 5.3: Mercury: Sample Django URLs Snippet.

```
from django.views.generic.base import TemplateView

class AllApplicationsView(TemplateView):
    template_name = "applications/allapps.html"

class ApplicationDetailView(TemplateView):
    template_name = "applications/appdetails.html"
```

Listing 5.4: Mercury: Sample Django View Handler.

```
<li class='span3'>
    <% if (hasActions) { %>
    <div class='image-div'>
    <% } %>
    <a href='<%= app.view_url %>?ref=<%= baseURL%>'
     rel='tab' class='thumbnail'>
        <img src=<%= app.icon_square_url %>
         onerror='imageError(this);' alt='No image available'>
    </a>
    <% if (hasActions) { %>
        <input type='checkbox' id='<%= app.id %>'
         class='check-item thmbnail-checkbx select-app-for-action'>
    </div>
    <% } %>
    <a href='<%= app.view_url %>?ref=<%= baseURL%>'
     rel='tab' class='black-link'>
        <p class='thumbnail-name ellipsis'><%= app.name %></p>
    </a>
    <div id='<%= appReview %>' class='review-div'></div>
</li>
```

Listing 5.5: Mercury: Sample Django Template Snippet.

within the Django models. Django Allauth (version 0.10.1) [Pen13] extends Django's built-in user support to integrate with third party authentication systems (e.g. Facebook, Google, OpenID). The Django REST framework (version 2.3.3) [Chr13] allows Django views to be used to provide a REST API and Django Cors Headers (version 0.12) [Yiu13] provides support for CORS (Cross-Origin Resource Sharing) headers, allowing both same-domain and cross-domain GET requests to the REST API. Finally, the Django Filter plugin (version 0.6) [Ale13] provides a set of filters to be used in Django views; this package is used by the Django REST framework to support data retrieval operations.

At the client side Mercury uses HTML 4, CSS and JavaScript as the underlying technologies; these all represent popular choices for the web platform. We use two front-end frameworks to improve the web UI implementation process. JQuery (version 2.0.3) [jF13] provides shortcut JavaScript operations for DOM manipulation, event handling and calls to the Mercury API using AJAX. The web UI framework Bootstrap [OT13b] (version 2.3.2) provides HTML and CSS templates for interface components including menu bars and buttons. Both JQuery and Bootstrap are open source projects with large user bases.

As in the Yarely development, a number of passive testing tools were used to ensure resilient, robust code: Pyflakes [Pyt13d] and pep8 [Pyt13c] were used as in Yarely and McCabe complexity was also calculated – all three tests were executed using the Flake8 source code checker [Pyt13a]. Static code analysis was also conducted on the JavaScript UI code using JSHint [Kov13]. Active tests were written for the Python backend using the `django.test` module, a Django adaptation of the standard Python `unittest` module.

Code was versioned using the Kiln [Fog13a] (Mercurial [Mac13]) version control system. In addition, a Buildbot [Bui13] continuous integration system connected to web hooks allowed us to trigger our passive Python and JavaScript tests following each commit to the code repositories; all tests executed on an Ubuntu 13.04 development server. Execution of Python unit tests on committed code was achieved using Django Discover Runner (version 0.4) [Lei13] – this plugin is integrated with Django from Django 1.6 onwards. Finally, as in the Yarely development process, task management and code reviews were managed using Fogbugz [Fog13b].

5.4.2 Implementation

The implementation of the application store was largely conducted by Mateusz Mikusz and Miriam Greis and is reported in their diploma theses [Gre13, Mik13].

In accordance with the original design, the Mercury implementation includes two interfaces, both contained within the same Django project. The RESTful API interface is implemented entirely in Django (Python). The web user interface (UI) is implemented as a combination of Django (Python) models, views and templates together with JavaScript and CSS. A breakdown by programming language is given in Table 5.2.

A Django website is composed of a *project* containing a number of *apps*. In Django terminology, the project represents the complete "collection of configuration and applications for a... Web site" [Dja13b]. An app is a small, potentially reusable, library that is "designed to represent a single aspect of a project" [GR13, p. 23]. Each Django app is a Python package within the top-level project Python

	Files	**Lines of Code**
Python	145	6809
JavaScript	17	2736
Templates and HTML	45	1796
CSS	7	607

Table 5.2: Breakdown of markup, programming and stylesheet languages used in Mercury. Framework code (e.g. Bootstrap, JQuery) is not included in these counts.

package (textttmercury). The Mercury project is divided into thirteen apps/ subpackages. Figure 5.3 shows the project /application hierarchy; an additional subpackage `mercury.mercury` is created by Django to contain project-wide configuration.

The thirteen Django apps are as follows:

users Extends Django's built-in User account system to allow individuals to log into Mercury to manage displays and applications. The users application provides functionality to allow a user to connect their account to a `DevelopmentCompany` through a `DeveloperProfile`; applications published by the user will be associated with this `DevelopmentCompany`.

applications Provides functionality to describe and store different types of public display applications. The `applications` app provides support for two distinct kinds of applications.

Slideshow Applications (implemented as DropBox folders) allow support for the current common use case in digital signage in which displays typically cycle through a set of simple media items (images, video etc.). By contrast, *Web Applications* represent the trend towards more complex media and the current typical method for developing these.

Application descriptions may be retrieved as JavaScript Object Notation (JSON) or as a Content Descriptor Set (CDS).

apikeys Provides mechanisms for generating and storing third-party API keys to interact with other services (e.g. Dropbox).

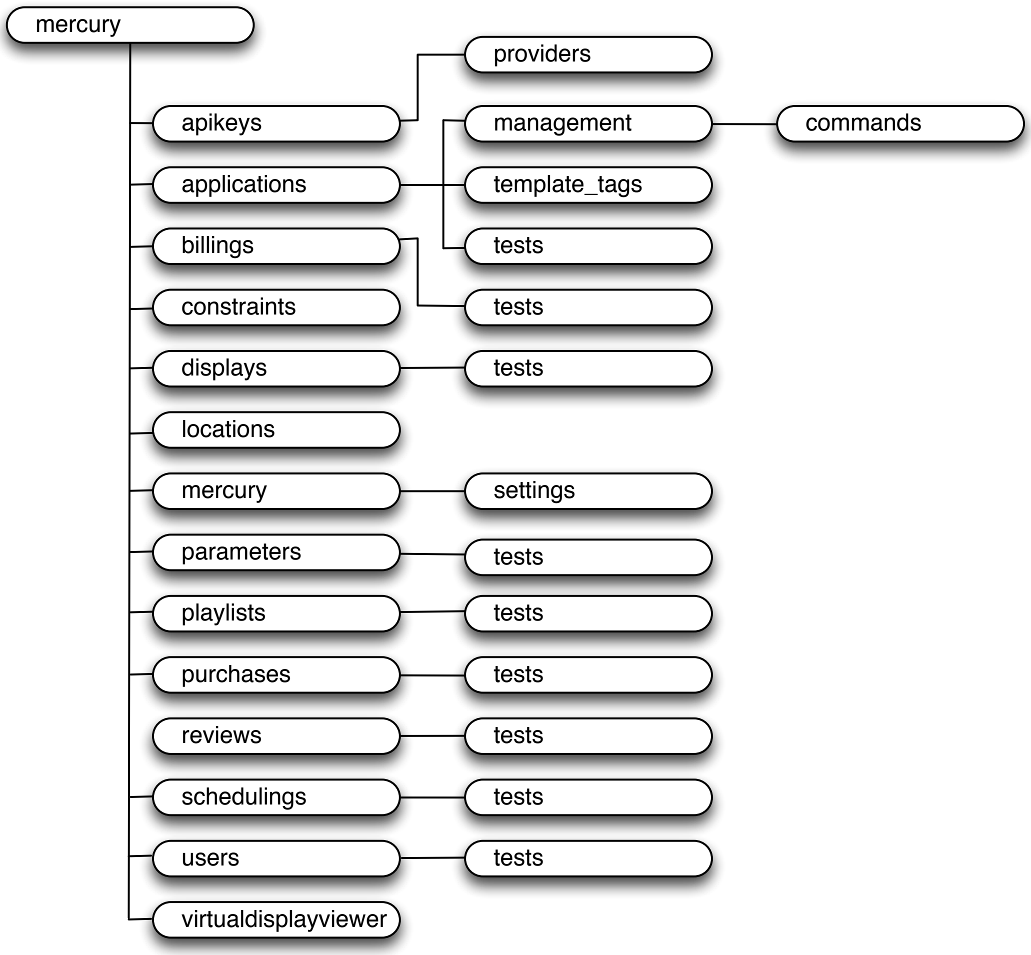


Figure 5.3: Python package and subpackage hierarchy for the Mercury implementation.

billings Allows the representation of different billing models (e.g. one-off purchase, monthly- subscription).

constraints Provides mechanisms to store and retrieve restrictions on the availability of an application or display.

parameters Allows applications to accept user- or location- specific parameters for customisation.

playlists Provides functionality to describe and store ordered groups of public display applications. Playlist descriptions may be retrieved as JSON or as a CDS.

purchases Provides a mechanism for representing and verifying application purchases and their validity (e.g. whether a subscription has ended).

reviews Allows a user to contribute textual or numeric ratings of their experiences with a purchased application.

displays Provides representation of physical display hardware configurations.

locations Provides mechanisms for describing the locations of physical display hardware.

schedulings Allows applications and playlists to be stored as an ordered schedule for playback at a display. Schedule descriptions may be retrieved as JSON or as a CDS.

virtualdisplayviewer Provides a simple web media player that cycles through all items scheduled to a display.

5.4.2.1 Data storage

Entities from our original entity-relation design (Figure 4.9, page 176) are represented as Django models contained within the thirteen apps. The Django object-relational mapper stores model instances in a MySQL database.

The Django models match closely to our original entity-relations design [Table 5.3]. Ten of our twelve entities are implemented as custom data models (of the remaining two entities, the User entity is mapped to Django's builtin User model, and Play Records are left for future work). We provide twenty-three custom models in addition to those specified by our original entity-relation design of which six are specialisations (sub-classes) of existing entities and one maps to a relation in our entity-relation design. The full set of models is shown in Figure 5.4.

Designed Entity	Implemented Model
Application	`mercury.applications.Application`
Billing Model	`mercury.billings.BillingModel`
Category	`mercury.applications.Category`
Developer Profile	`mercury.users.DeveloperProfile`
Development Company	`mercury.users.DevelopmentCompany`
Display	`mercury.displays.DisplayModel`
Display Group	`mercury.displays.DisplayGroup`
Play Record	Not implemented
Playlist	`mercury.playlists.PlaylistModel`
Purchase Agreement	`mercury.purchases.PurchaseModel`
Review	`mercury.reviews.Review`
User	`django.contrib.auth.models.User`

Table 5.3: Mercury: Mapping from designed entity's to implemented data models.

5.4.2.2 User Interface

The user interface remains close to the original design (shown in the previous Chapter, Figures 4.10 to 4.13). We retain the designed structure of a layout composed of sidebar, top menu and main frame. We use Bootstrap's `navbar-fixed-top` and `sidebar-menu` classes to create the top menu and sidebar respectively. Contents of the main frame are laid out using Bootstrap's grid system. A realisation of our original workflows as screenshots from the web user interface (UI) implementation are shown in Figures 5.5 to 5.7. Functionality for the web UI is provided through calls to the RESTful API made using Ajax. The RESTful API is described in Section 5.4.2.3. Ajax calls are made using the JQuery library.

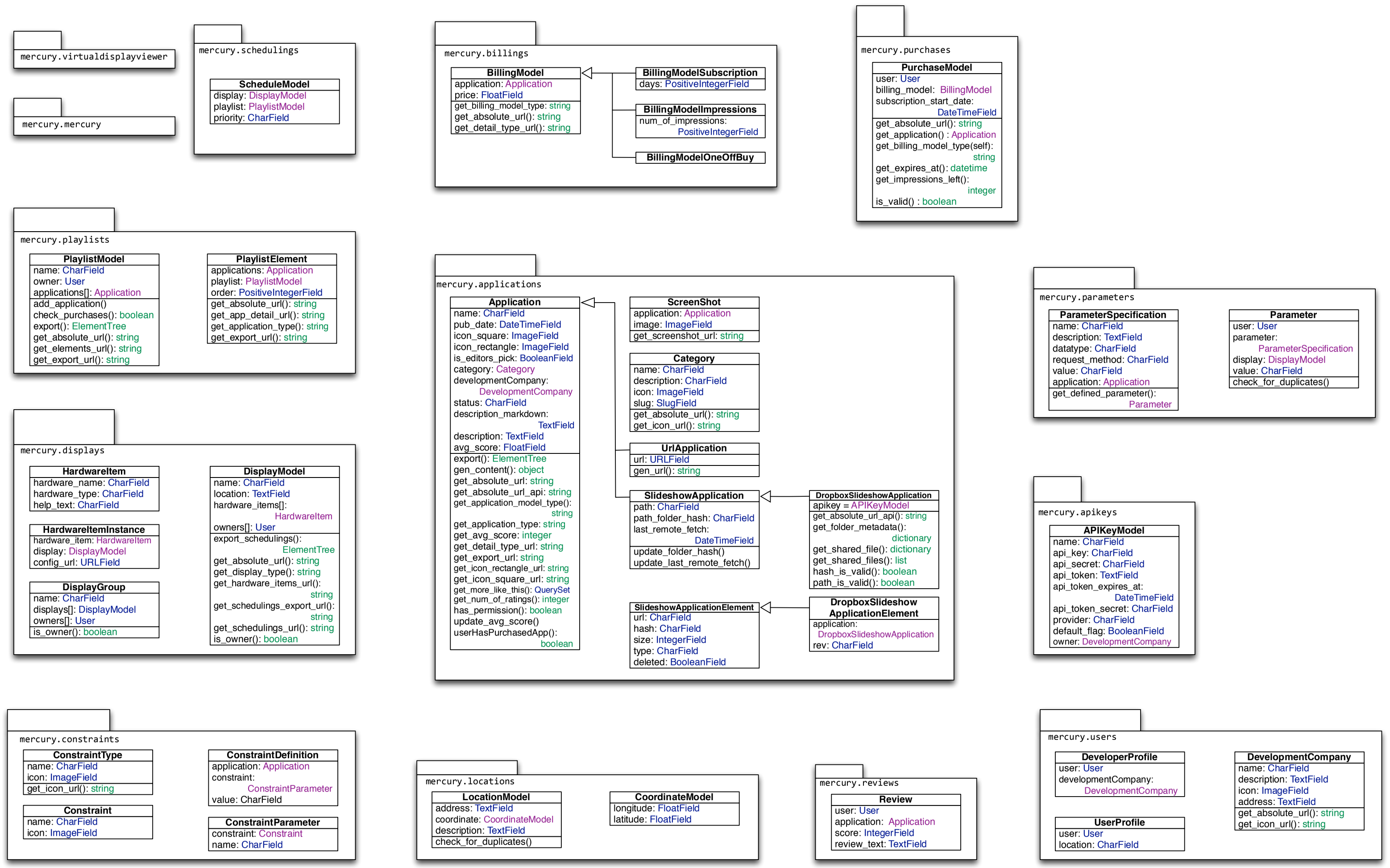


Figure 5.4: Django models (database objects) for the Mercury implementation.

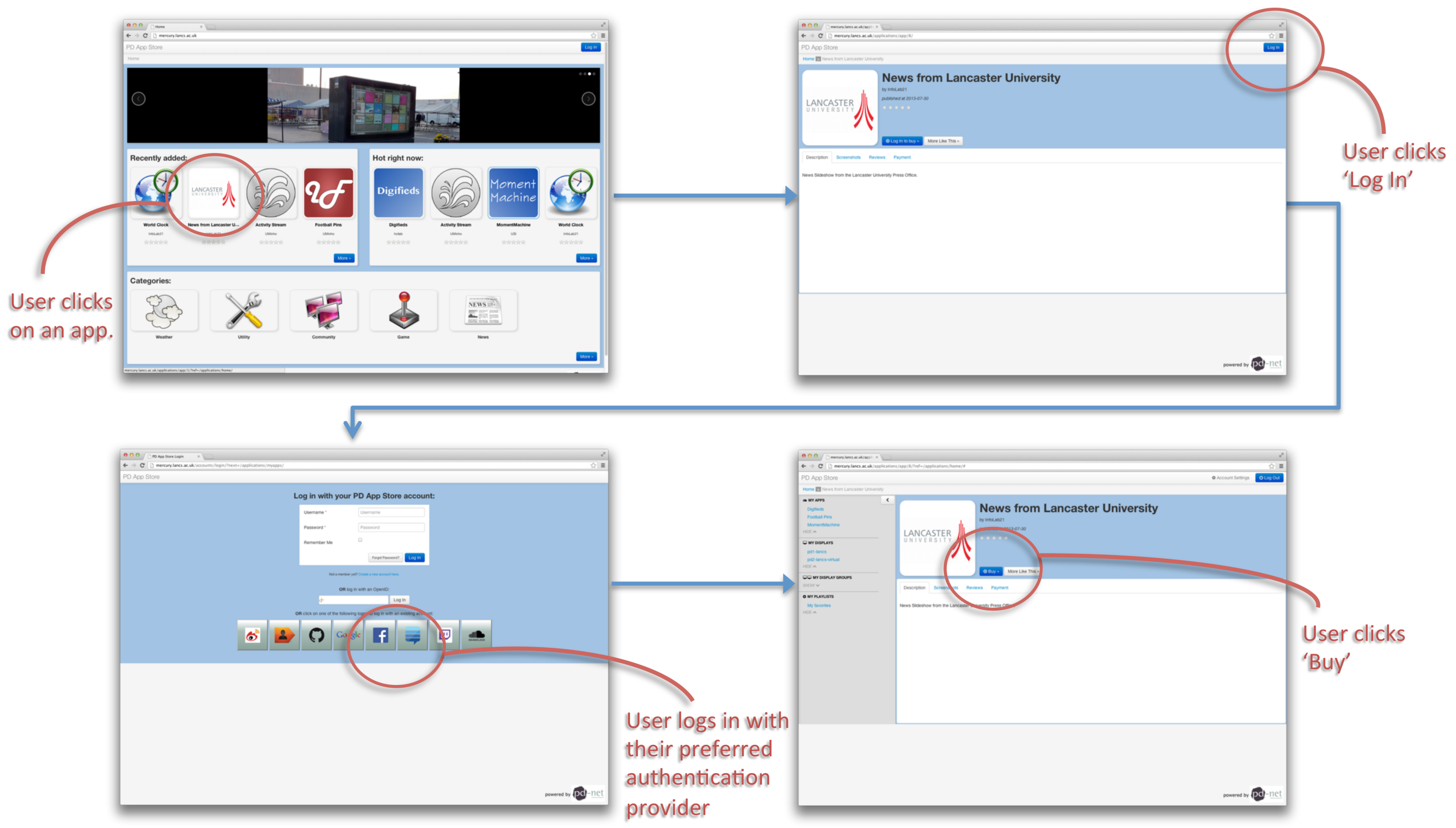

User clicks on an app.
User clicks 'Log In'
User logs in with their preferred authentication provider
User clicks 'Buy'
PD App Store
News from Lancaster University
Log in with your PD App Store account:

Figure 5.5: Sample Mercury web interface workflow. This workflow shows how a user can browse the application store, view application details, log in, and purchase an application. Based on the design in Figure 4.11.

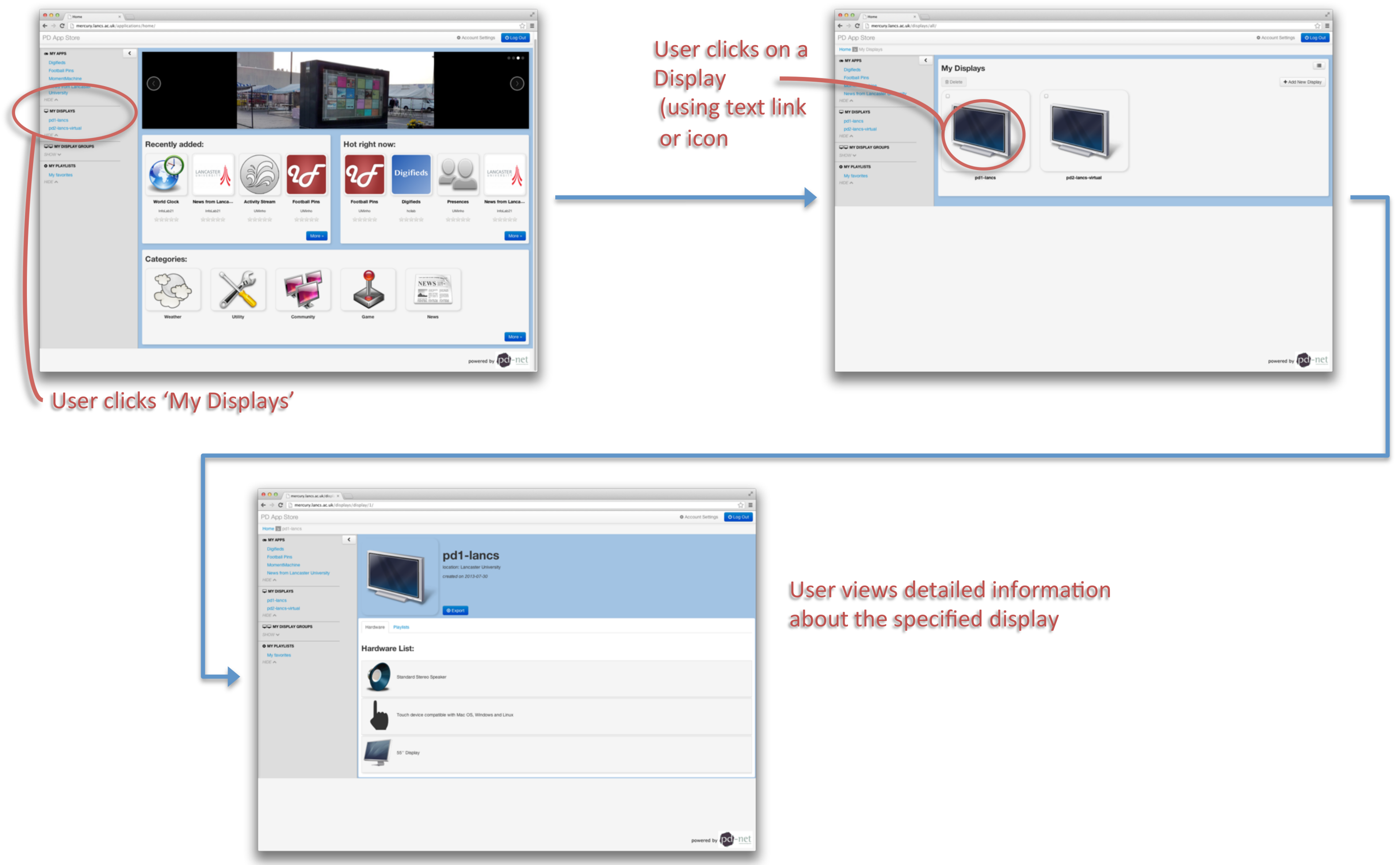

User clicks 'My Displays'
User clicks on a Display (using text link or icon
User views detailed information about the specified display
My Displays
pd1-lancs
pd2-lancs-virtual
Recently added:
Hot right now:
Categories:
Hardware List:
Standard Stereo Speaker
Touch device compatible with Mac OS, Windows and Linux
55" Display

Figure 5.6: Sample Mercury web interface workflow. This workflow shows how a display owner can browse the display hardware that they have registered with the application Store. Based on the design in Figure 4.12.

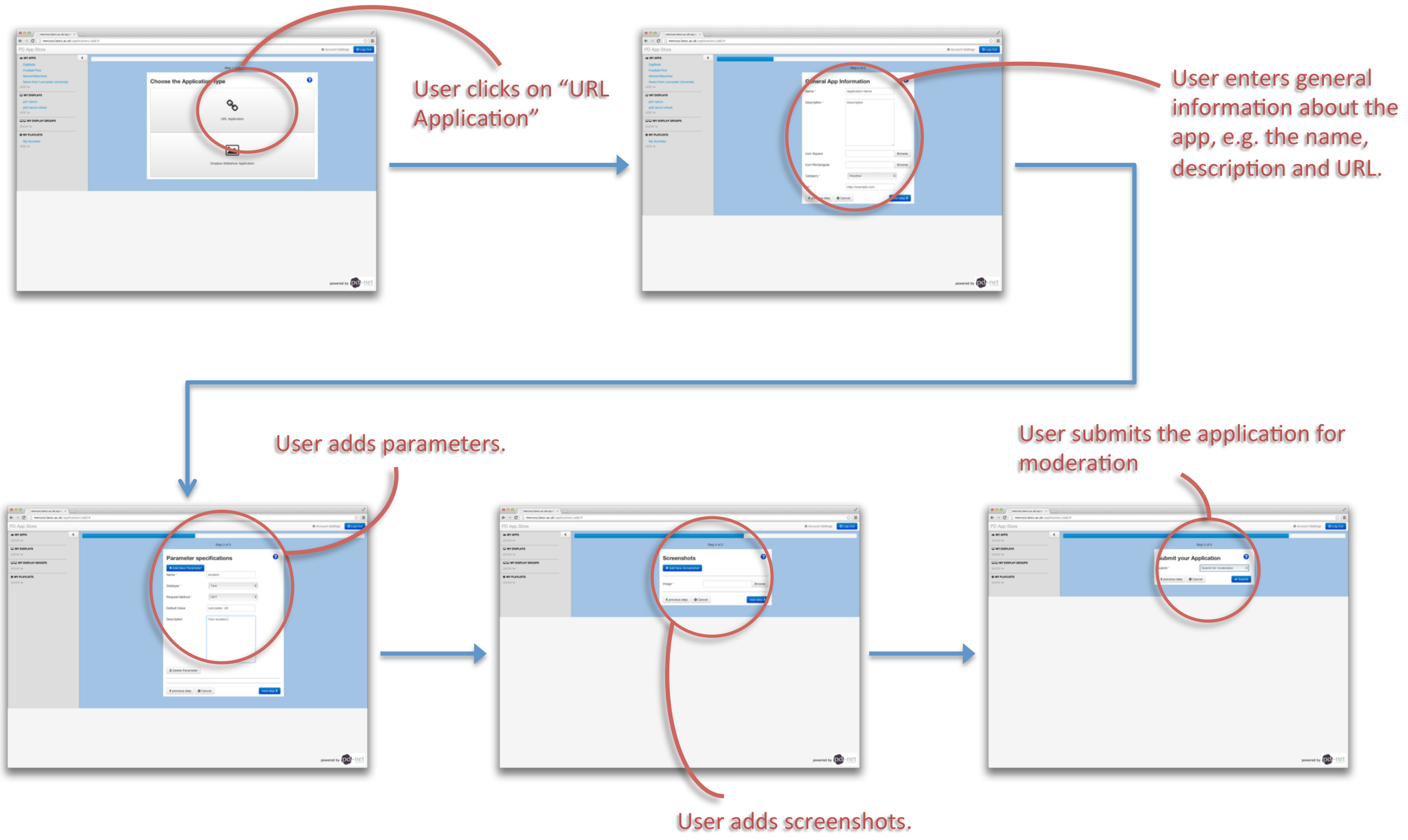

User clicks on "URL Application"
User enters general information about the app, e.g. the name, description and URL.
User adds parameters.
User adds screenshots.
User submits the application for moderation
Choose the Application Type
General App Information
Parameter specifications
Screenshots
Submit your Application

Figure 5.7: Sample Mercury web interface workflow. This workflow shows how an application developer (content provider) can add a new application to the application store. Based on the design in Figure 4.11.

5.4.2.3 RESTful APIs

The majority of functionality provided by the web user interface is made available through RESTful API calls. This RESTful API also allows other tools to access application store data and operations.

We provide eight resources as part of the RESTful API:

applications This resource provides listings of and details for Application objects. Application objects may be filtered and/or searched to limit results. This resource can be accessed at `/api/applications/`.

billings This resource provides listings of and details for BillingModel objects. This resource can be accessed at `/api/billings/`.

development_companies This resource provides listings of and details for DevelopmentCompany objects and allows an authenticated user to create new DevelopmentCompany objects. This resource can be accessed at `/api/users/developmentcompanies/`.

displays This resource allows an authenticated user to access listings of and details for all associated DisplayModel objects. This resource can be accessed at `/api/displays/`.

playlists This resource allows an authenticated user to listing, detail and edit all associated PlaylistModel objects. This resource can be accessed at `/api/playlists/`.

purchases This resource allows an authenticated user to create and view PurchaseModel objects. This resource can be accessed at `/api/purchases/`.

reviews This resource provides listings of and details for Review objects. Review objects may be filtered and/or searched to limit results. Authenticated users may also create new Review objects through this resource. This resource can be accessed at `/api/reviews/`.

schedulings This resource provides listings of and details for the Playlist objects that have been schooled to the specified Display(s). This resource can be accessed at `/api/schedulings/`.

A sample API request and response is shown in Figure 5.8.

5.5 Tacita: Supporting Personalisation/Appropriation Requests

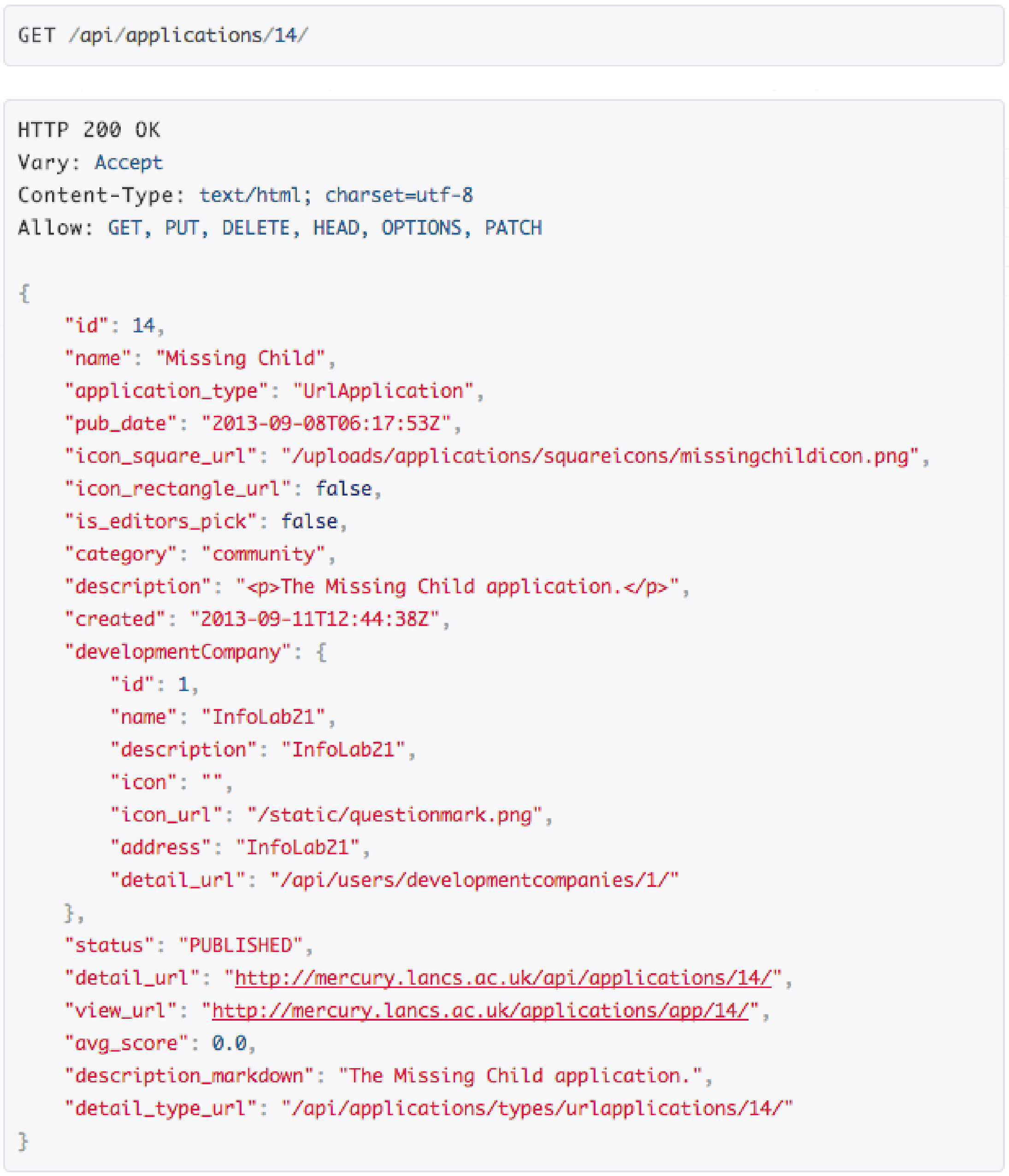
5.5.1 Platform and Tools

Tacita is comprised of a mobile client, optional map provider, and a set of personalised application services. The mobile client was implemented by Thomas Kubitza and Christopher Winstanley and the map provider by Thomas Kubitza. Our implementation has focussed on the integration of these components, and collaboration with the authors in order to produce a set of six personalised application services.

In order to provide good coverage of current mobile devices, implementations of the Tacita mobile client are provided for both Android 4.2 Jelly Bean and iOS 7. The Android client is developed in Java and XML whilst the iOS application is written in Objective-C.

On both mobile platforms the clients use multiple methods for detecting proximity to displays. This is achieved by using Android's Location APIs [AOSP13] and the iOS Core Location Framework [App13] to abstract over underlying location technologies (GPS, WiFi fingerprinting and cell tower triangulation). Both mobile clients also use Bluetooth monitoring to detect when a viewer comes into the propagation zone for a display. These libraries allow access to all available location technologies in an energy-efficient manner and also provide simple libraries for calculating location intersections and proximity. Both mobile client implementations also support explicit triggering through use of a QR code. QR codes provide a visible cue to display viewers and are well-suited to quick interpretation by personal mobile devices.

The Tacita map provider is implemented in PHP 5 [PG13c] and uses Zend Framework 2 [Zen13]. Map data responses are issued in XML due to its human-readability, considerable programming language support, and common use for data representation in web services.



```
GET /api/applications/14/
```

```
HTTP 200 OK
Vary: Accept
Content-Type: text/html; charset=utf-8
Allow: GET, PUT, DELETE, HEAD, OPTIONS, PATCH

{
    "id": 14,
    "name": "Missing Child",
    "application_type": "UrlApplication",
    "pub_date": "2013-09-08T06:17:53Z",
    "icon_square_url": "/uploads/applications/squareicons/missingchildicon.png",
    "icon_rectangle_url": false,
    "is_editors_pick": false,
    "category": "community",
    "description": "<p>The Missing Child application.</p>",
    "created": "2013-09-11T12:44:38Z",
    "developmentCompany": {
        "id": 1,
        "name": "InfoLab21",
        "description": "InfoLab21",
        "icon": "",
        "icon_url": "/static/questionmark.png",
        "address": "InfoLab21",
        "detail_url": "/api/users/developmentcompanies/1/"
    },
    "status": "PUBLISHED",
    "detail_url": "http://mercury.lancs.ac.uk/api/applications/14/",
    "view_url": "http://mercury.lancs.ac.uk/applications/app/14/",
    "avg_score": 0.0,
    "description_markdown": "The Missing Child application.",
    "detail_type_url": "/api/applications/types/urlapplications/14/"
}
```

Figure 5.8: Sample Mercury API request and response. In this call, a GET request to the applications resource on `/api/applications/<application_id>/` results in a HTTP response containing a JSON representation of the specified application.

A total of six personalisable display applications for use with Tacita. Five applications (*PD News*, *World Clock*, *PD Weather*, *ubiVM*, *newsBounce*) were implemented as dynamic websites, allowing them to be shown on a wide range of display hardware and software systems; the sixth application (*native-VNC*) was implemented as a web service that bridged from Internet Suspend/Resume [KS02] to the Yarely HTTP server sensor handler. Technologies used across the applications include PHP 5 [PG13c], Zend Framework 2 [Zen13], the PEAR HTML_Template_IT templates [PG13a], Python 2.7 [Pyt13e], the Tornado web framework [Fac13], SQLite [Hip13], SPARQL [W3C13], HTML 5, CSS, web sockets, JavaScript, and Ajax long polling.

A number of third-party APIs are used to source data for the personalised application services. Our *World Clock* and *PD Weather* applications both use the Google Maps geocoding [Goo13a] and timezone [Goo13b] APIs. The *PD Weather* application also used the Google Weather API until its removal in August 2012. The *PD News* application uses the Google news API [Goo12] to find news stories. Similarly, our *newsBounce* application uses the BBC Content, Locations and Juicer APIs [BBC13b] to source its news stories. We use HTML 5 widgets from ClockLink.com [Pac13] to show times in the *World Clock* application. Finally, the *ubiVM* application renders remote desktop sessions using a Java-based VNC client provided as a Jar file from TightVNC [1], an open source VNC client, whilst our *native-VNC* application uses Yarely's platform-specific VNC renderers.

5.5.2 Implementation

Tacita is comprised of a client application for viewers' mobile devices, an optional map provision service, and a set of personalised display applications. In this thesis, we integrate the existing implementations of the mobile client by Thomas Kubitza and Christopher Winstanley and the map provider by Thomas Kubitza. with our other architectural components, and develop a set of personalised application services used to demonstrate the integration into our overall architecture.

The following sections describe the implementation of each of Tacita's components, including our personalised application services.

[1] http://www.tightvnc.com/

5.5.2.1 Mobile Client

The Tacita mobile client allows viewers to control the personalisation of public displays in their environment. In order to provide good coverage of current personal mobile devices, the implementation of the mobile client includes two distinct applications – one for Android devices and one for iOS devices.

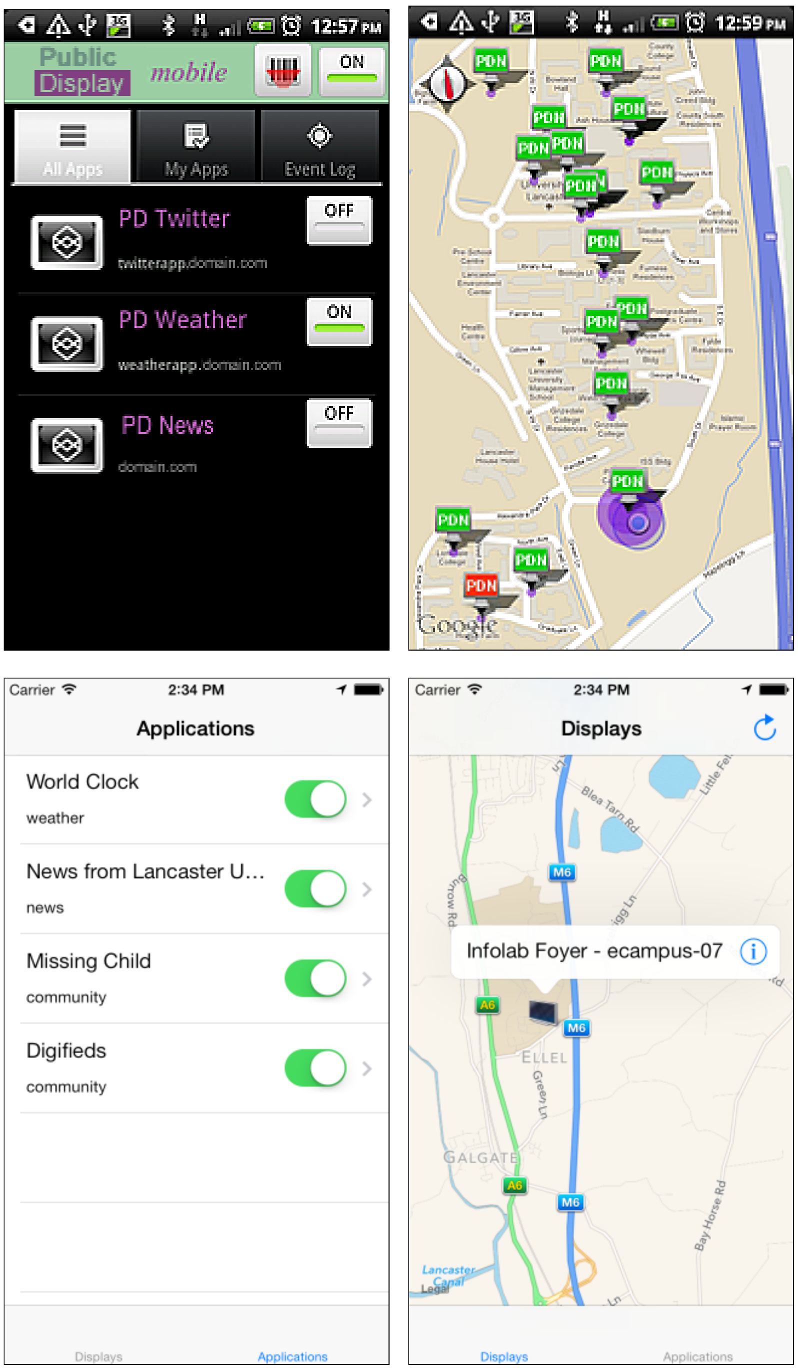
The central function of the client is to maintain a list of applications to which the user is subscribed and then, using its knowledge of displays and their capabilities, to trigger applications when the user enters a *trigger zone* around the display. Both mobile client implementations use circle sectors (i.e. radius, centre point, and start/end number of degrees) to describe location-based trigger zones around the displays.

The Tacita mobile client allows users to view nearby displays (including their trigger zones) and see all available applications [Figure 5.9]. Users can select applications to enable and, for each selected application, may also set parameters to customise display output (e.g. by entering search terms to filter news, location information to customise weather or time, or authentication credentials to personalise an application based on some pre-existing profile). These parameters are passed to the selected applications with information about the users' nearby displays.

The mobile clients support two methods of triggering a personalisation request:

- **Implicit triggering** uses the location APIs and Bluetooth sensing to trigger application requests. Location tracking and Bluetooth scanning can run as a background service allowing a user to passively personalise displays in addition to making active choices about the displays they would like to appropriate. Location tracking and Bluetooth scanning services can be customised or deactivated to conserve power.

- **Explicit triggering:** allows a user to explicitly request a public displays' foreground independently of their location tracking technologies by scanning a QR-code provided by the display. For example, this could allow a user to appropriate a display even when operating their mobile device in a power-saving (i.e. non-location-tracking) mode.


12:57 PM

Public Display mobile ON

All Apps | My Apps | Event Log

PD Twitter OFF
twitterapp.domain.com

PD Weather ON
weatherapp.domain.com

PD News OFF
domain.com

12:59 PM

Carrier 2:34 PM

Applications

World Clock
weather

News from Lancaster U…
news

Missing Child
community

Digifieds
community

Displays | Applications

Carrier 2:34 PM

Displays

Infolab Foyer - ecampus-07

Displays | Applications

Figure 5.9: Tacita mobile client UI on Android (top) and iOS (bottom). The screenshots show the interface for viewing available applications (left) and nearby displays (right).

5.5.2.2 Display Infrastructure and Presence Confirmation

If a display requires that a viewer provides presence confirmation in order to have their personalised content scheduled, then that display must provide a mechanism for distributing some form of presence token to nearby personal mobile devices.

Implementation of this feature is currently provided through use of QR-code generation. A new QR-code is periodically generated and overlaid on web content through a custom web application. The generated QR-code acts as both capability announcement and presence token.

5.5.2.3 Map Provider

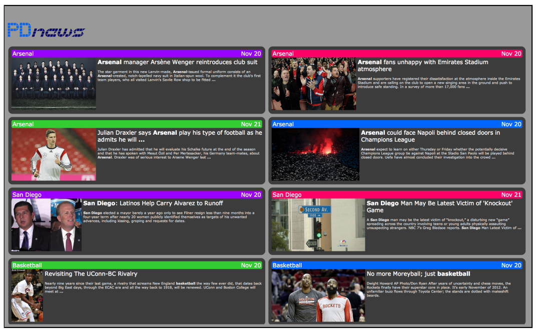
The map provider is an optional component capable of caching display capability announcements in order to serve collated sets of announcements to personal mobile devices. The map provider is implemented as a web service that supplies mobile clients with an XML file containing capability announcements for displays within a specified geographic region [Listing 5.6]. To reduce transfer time, the map XML is gzip compressed between the web service and mobile devices.

5.5.2.4 Personalised Applications

In order to demonstrate Tacita and evaluate its integration into our overall architecture, we have collaborated with the mobile client authors to create a set of six applications with support for viewer personalisation [Figure 5.10]. These include:

News applications *PD News* and *newsBounce* build on existing usages of pervasive displays to show news stories. *PD News* [Figure 5.10a] allows viewers to express search terms that are used to filter news stories shown at the display. By contrast, *newsBounce* [Figure 5.10b] allows a user to follow a set of related stories on a particular topic – every new display encountered will show a different story from the pool of unread stories within the topic.

PD Weather [Figure 5.10d] that enables viewers to express preferences for the specific locations about which weather information is shown.



PDnews

Arsenal Nov 20
Arsenal manager Arsène Wenger reintroduces club suit
The star garment in this new Lanvin-made, **Arsenal**-issued formal uniform consists of an **Arsenal**-crested, notch-lapelled navy suit in Italian-spun wool. To complement it the club's first team players, who all visited Lanvin's Savile Row shop to be fitted ...

Arsenal Nov 20
Arsenal fans unhappy with Emirates Stadium atmosphere
Arsenal supporters have registered their dissatisfaction at the atmosphere inside the Emirates Stadium and are calling on the club to open a new singing area in the ground and push to introduce safe standing. In a survey of more than 17,000 fans ...

Arsenal Nov 21
Julian Draxler says **Arsenal** play his type of football as he admits he will ...
Julian Draxler has admitted that he will evaluate his Schalke future at the end of the season and that he has spoken with Mesut Özil and Per Mertesacker, his Germany team-mates, about **Arsenal**. Draxler was of serious interest to Arsene Wenger last ...

Arsenal Nov 20
Arsenal could face Napoli behind closed doors in Champions League
Arsenal expect to learn on either Thursday or Friday whether the potentially decisive Champions League group tie against Napoli at the Stadio San Paolo will be played behind closed doors. Uefa have almost concluded their investigation into the crowd ...

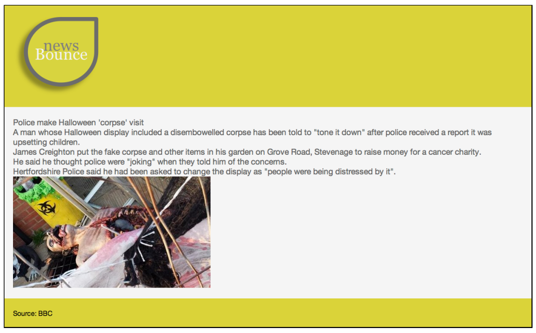
San Diego Nov 20
San Diego: Latinos Help Carry Alvarez to Runoff
San Diego elected a mayor barely a year ago only to see Filner resign less than nine months into a four-year term after nearly 20 women publicly identified themselves as targets of his unwanted advances, including kissing, groping and requests for dates.

San Diego Nov 21
San Diego Man May Be Latest Victim of 'Knockout' Game
A **San Diego** man may be the latest victim of "knockout," a disturbing new "game" spreading across the country involving teens or young adults physically assaulting unsuspecting strangers. NBC 7's Greg Bledsoe reports. **San Diego** Man Latest Victim of ...

Basketball Nov 20
Revisiting The UConn-BC Rivalry
Nearly nine years since their last game, a rivalry that screams New England **basketball** the way few ever did, that dates back beyond Big East days, through the ECAC era and all the way back to 1918, will be renewed. UConn and Boston College will meet at ...

Basketball Nov 20
No more Moreyball; just **basketball**
Dwight Howard AP Photo/Don Ryan After years of uncertainty and chess moves, the Rockets finally have their superstar core in place. It's early November of 2012. An unfamiliar buzz flows through Toyota Center; the stands are dotted with makeshift beards.

(a) *PD News*

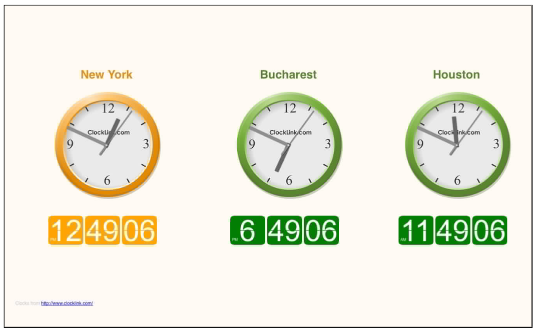


news Bounce

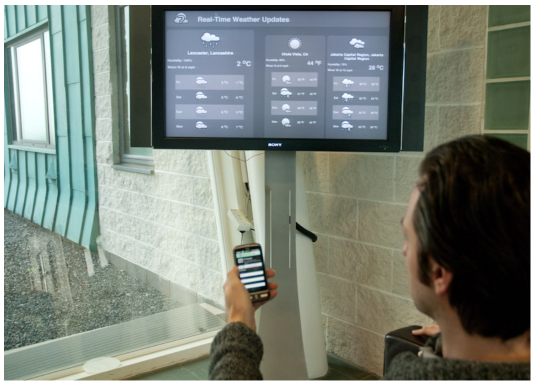
Police make Halloween 'corpse' visit
A man whose Halloween display included a disembowelled corpse has been told to "tone it down" after police received a report it was upsetting children.
James Creighton put the fake corpse and other items in his garden on Grove Road, Stevenage to raise money for a cancer charity.
He said he thought police were "joking" when they told him of the concerns.
Hertfordshire Police said he had been asked to change the display as "people were being distressed by it".

Source: BBC

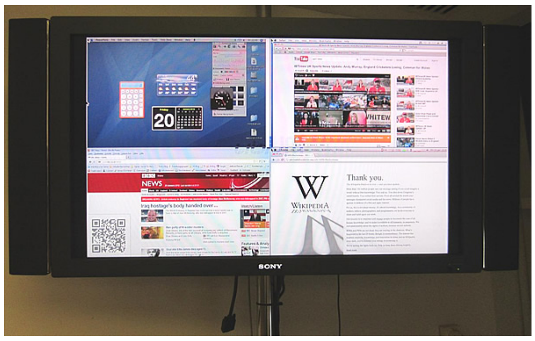
(b) *newsBounce*



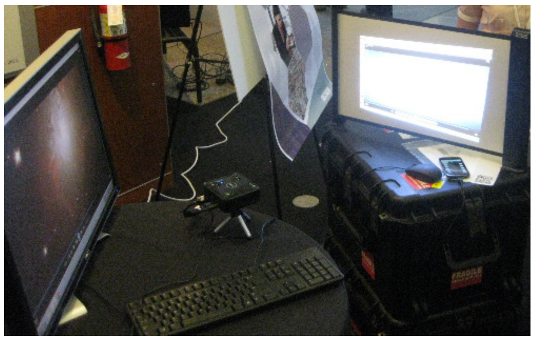
(c) *World Clock*



(d) *PD Weather*



(e) *ubiVM*



(f) *native-VNC*

Figure 5.10: Tacita display application interfaces (DIs): screenshots of *PD News*, *newsBounce* and *World Clock*; in-situ photographs of *PD Weather*, *ubiVM* and *native-VNC*.

```xml
<displays>
  <display>
    <uri>exampledisplay.testdomain.edu</uri>
    <name>An Example Display</name>
    <owner><name>Testdomain inc</name></owner>
    <location>
      <lat>25.0</lat><lon>71.0</lon><alt>0</alt>
      <radius>200</radius>
      <orientation>N7</orientation>
      <view-angle-left>0</view-angle-left>
      <view-angle-right>360</view-angle-right>
    </location>
    <capabilities>
      <apps>
        <app>
          <uri>news.newsprovider.com</uri>
          <name>PD News</name>
          <description>Personalisable News</description>
          <provider><name>News Provider</name></provider>
          <capabilities>
            <display-foreground-api>
              <url>http://news.newsprovider.com/api/</url>
              <parameters>
                <parameter pname="tag" type="text"
                    required="true"/>
              </parameters>
            </display-foreground-api>
          </capabilities>
        </app>
      </apps>
      <presence-token-broadcast enabled="true"/>
      <bluetooth-broadcast>
        <mac-id>00:1e:3d:f8:69:21</mac-id>
      </bluetooth-broadcast>
    </capabilities>
  </display>
</displays>
```

Listing 5.6: Sample XML output from the Tacita Map Provider.

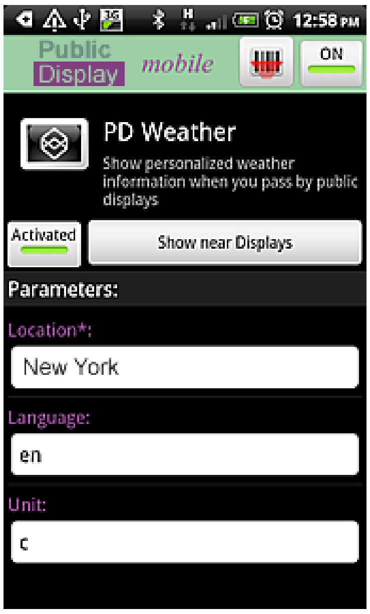
World Clock [Figure 5.10c] that builds on recent research that suggests that viewers now have renewed interest in ambient time displays [FDE12, Sta13]. This application enables users to request that local time and at least one additional timezone is shown at the display.

Remote desktop applications *ubiVM* and *native-VNC* that facilitate highly-customised personalised displays by enabling users to request that a screen shows the display output of a VNC connection to a remote computer. *ubiVM* [Figure 5.10e] allows viewers to connect to any physical or virtual computer running a VNC server whilst *native-VNC* [Figure 5.10f] focuses on virtual machine support, allowing viewers to connect to a VM located local to the display or in the wider Internet; this latter approach can allow resource-intensive applications to overcome performance issues associated with network latency and remote execution.

Each of our applications are built using a common architecture. Each application takes a set of parameters supplied by the mobile device (e.g. Figure 5.11) and uses those parameters to generate output suitable for showing at a public display. The applications each provide two distinct interfaces [Figure 5.12]:

1. a *Request Interface* (RI) that allows mobile devices to submit requests for a display to be personalised.

2. a *Display Interface* (DI) that is accessed by public displays and that presents application content in a format suited to the known resolution of the display.



PD Weather
Show personalized weather information when you pass by public displays

Parameters:

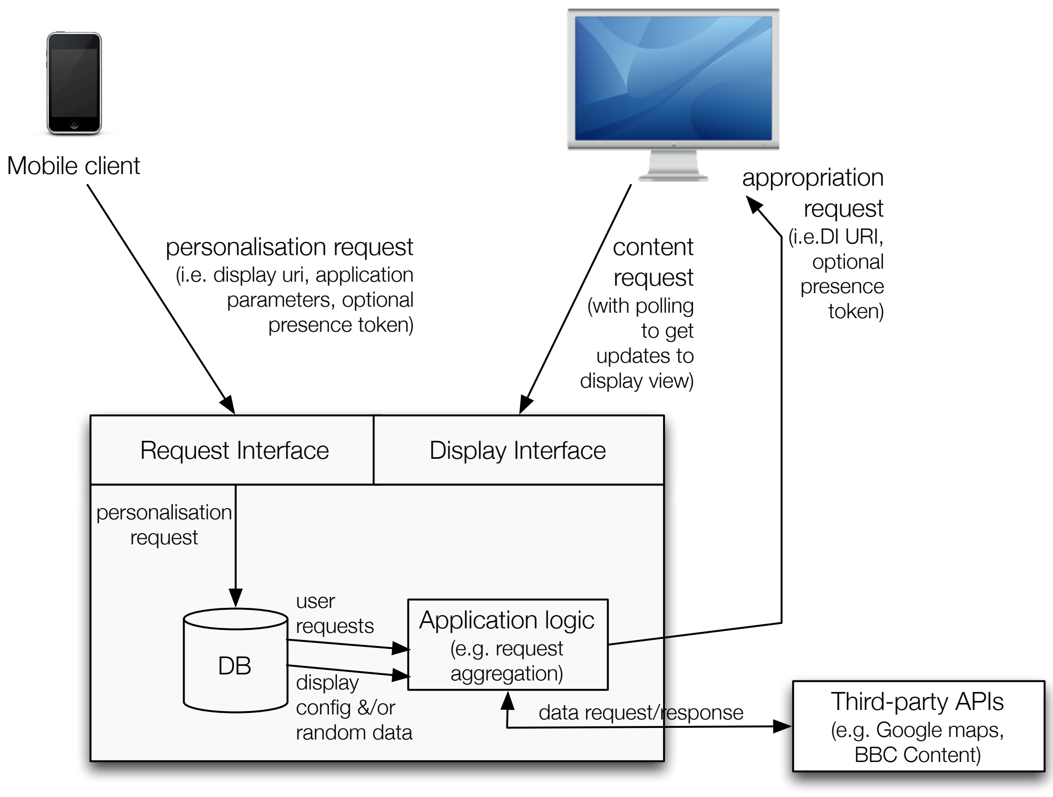
Location*:
New York

Language:
en

Unit:
c

Figure 5.11: Specifying parameters for the *PD Weather* application using the Tacita mobile client (Android) UI.

Each applications maintains a local database of the user requests it has received – each request will include a display URI, any presence



Mobile client
personalisation request
(i.e. display uri, application parameters, optional presence token)
content request
(with polling to get updates to display view)
appropriation request
(i.e.DI URI, optional presence token)
Request Interface
Display Interface
personalisation request
user requests
DB
display config &/or random data
Application logic
(e.g. request aggregation)
data request/response
Third-party APIs
(e.g. Google maps, BBC Content)

Figure 5.12: Tacita: Generic architecture for the six personalised display applications. Each application features a Request Interface (RI) that accepts incoming personalisation requests from mobile clients, and a Display Interface (DI) that provides content to be shown at a display.

token currently held for this display and parameters for this application. For example, the *PD Weather* application accepts a textual representation of the location for which a forecast should be fetched, the language forecast information should be shown in, and the preferred unit of measurement (celsius or Fahrenheit).

The application then aggregates all recent requests from a display together with any local display configuration (e.g. the *PD Weather* application will always show the local weather in addition to user-requested forecasts and so has a configuration option that specifies the area in which a display is located). At this stage, some applications also insert some random requests if the number of user requests is low – this makes it difficult to identify user-requested data when viewing the output on the display.

Once the application has collated the set of requests whose results will be shown at the display, it then uses third-party APIs to fetch data that will be used to generate the output made available on the DI. For example, The *PD Weather* application uses Google Maps' geocode services to resolve textual place names into appropriate locations then submits the results to a second Google API to fetch weather data.

Upon completion of any third-party API calls, the application makes an appropriation request to notify the display that it has content available to be shown. The appropriation request made to a display includes any presence token sent with the original user request and a URL or other identifier that directs this display to the DI.

In order to show the personalised application content, the display makes a content request to the DI. This results in a seamlessly updating representation of the displays visual output (e.g. by streaming VNC frames or using long-polling/web sockets to update objects within a web page). The resulting output is then shown at the display (as in Figure 5.10) – where randomly-generated requests have been inserted (e.g. in the *PD News*, *PD Weather* and *World Clock* applications) these become indistinguishable from user requests. This is an example of how applications can support plausible deniability, i.e. helping viewers to hide the fact that they are responsible for specific content.

5.6 Deployment

To date we have deployed Yarely on approximately thirty digital signage displays in day-to-day use on the Lancaster University campus, and to five displays at other universities worldwide. Yarely has also been demonstrated at several events. A publicly accessible instance of Mercury has been available for three months and contains applications created independently by researchers at a number of different institutions in addition to our own applications. Tacita remains an experimental system and has been demonstrated at a range of venues.

5.6.1 Yarely

Our first demonstration of Yarely was in November 2011, with a first deployment in February 2012. Our initial deployment consisted of four display nodes, one at each of four European universities.

We extended our deployment of Yarely onto the existing e-Campus hardware (approximately thirty Mac Mini computers with associated display hardware) in March 2012. This deployment of Yarely has been in continuous use on the Lancaster University campus since March 2012 as the dominant digital signage platform. Photographs showing sample locations in this deployment are shown in Figure 1.2 (page 13).

A typical display node in our deployment consists of a Mac Mini computer (2006/2007, Intel Core Duo 1.83GHz, 1GB DDR2 RAM, 80GB disk) running Mac OS X 10.6.8, and an associated Sony FWD40LX1 display. Approximately half of our deployed displays include speakers. One display in the deployment has a Sanyo projector in place of the Sony display and one has an outdoor display.

The above deployment (and early demonstration) runs the Mac OS X version of Yarely. The Mac OS X version has also been demonstrated at HotMobile 2012, PerDis 2012, Mobisys 2012, and a series of PD-NET [PD-13] project meetings. The Linux GNOME version of Yarely has been demonstrated at PerDis 2013 [XCDS13].

5.6.2 Mercury

A publicly accessible instance of Mercury has been available for five months at http://mercury.lancs.ac.uk/. The store currently contains ten applications including three applications from our own group and seven created from researchers at three other institutions.

Application	Developers	Description
Activity Stream	Universidade do Minho	Shows the recent history of a place.
Digifieds	Universität Stuttgart	An interactive digital classiffieds board.
Football Pins	Universidade do Minho	Shows viewers' preferred football images.
Missing Child	Lancaster University	Shows alerts to help find missing children.
Moments Gallery	Universitá della Svizzera italiana	Shows images from MomentMachine and Instagram.
Moment Machine	Universitá della Svizzera italiana	Allows viewers to take photos at a Display.
News from Lancaster University	Lancaster University	Shows recent press releases from Lancaster University.
Posters	Universidade do Minho	Shows posters left by recent visitors to a place.
Presences	Universidade do Minho	Shows recent visitors to a place.
World Clock	Lancaster University	Shows clocks for locations selected by current viewers.

Table 5.4: Mercury: Description of contributed applications. These applications are described further in the application authors' publications [AKB+11, ASKS13, JPS+12, JPSM13, MEM+13].

5.6.3 Tacita

Tacita and our personalised applications have been demonstrated at a number of venues.

Our first demonstration of the Android client and the personalised applications *PD Weather* and *PD News* took place at HotMobile 2012 [KCDL12]. Similar demonstrations were also made at MobiSys 2012 [CKDL12a] and Digital Futures 2012 [CKDL12b]. This demonstration used Bluetooth as the primary trigger mechanism.

The Tacita Android client and *native-VNC* personalised application service were demonstrated as part of an overall architecture for personalised virtual machine interaction on public displays at PerDis 2013 [XCDS13] – this demonstration combined both Tacita and Yarely with Internet Suspend/Resume [KS02]. Again, the demonstration used Bluetooth as the primary trigger mechanism.

Finally, the iOS version of Tacita and the *newsBounce* application were demonstrated at #newsHACK October 2013 [BB13]. This demonstration used proximity calculated from GPS as the primary trigger mechanism.

5.7 Summary

In this chapter we have described the implementation of: Yarely, a client-side scheduler and media player; Mercury, our application store for easy distribution of pervasive display applications; and the integration of Tacita, a mechanism for allowing users to generate appropriation requests and propagate them to the displays in their environment whilst maintaining some control over their personal data. We have also described the document format for our Content Descriptor Set (CDS) format that is used to represent content source and scheduling information in communications between Content Descriptor Factories (such as the application store) and scheduling and playback software (such as Yarely). The development of these four aspects of our overall architecture follows on from the design presented in Chapter 4.

This chapter has also provided details of deployments and demonstrations of the implemented components including a long-running installation of the Yarely media player – our current deployment spans four European countries and includes over thirty displays. In Chapter 6, we continue by discussing the integration of these implemented components and presenting an evaluation of the overall architecture and individual systems based on a mixture of deployment experiences, performance measures, user studies and reflective analysis.

Chapter 6

Evaluation

6.1 Overview

Chapter 5 described the implementation and deployment of an architecture to support user appropriation of pervasive displays. In this chapter we evaluate the architecture through a mixture of user studies, surveys and technical measurements.

We begin by considering the integration of the components described in Chapters 4 and 5. We then follow with a set of quantitative measurements for each of the three developed software systems: Yarely, Mercury, and Tacita. Following the quantitative measurements, we report a qualitative exploration of display owner and viewer attitudes to our architecture. We also reflect back on our deployment experiences and draw out a number of evaluation points from these.

For the final portion of the evaluation, we revisit our original set of requirements (first presented in Section 3.4.2) and analyse the integrated architecture with regard to these.

Portions of the work presented in this chapter have also been published in:

- Sarah Clinch, Nigel Davies, Adrian Friday, and Graham Clinch. Yarely – a software player for open pervasive display networks. In *Proceedings of the 2013 International Symposium on Pervasive Displays*, PerDis '13, New York, NY, USA, 2013. ACM [CDFC13].

- Nigel Davies, Marc Langheinrich, Sarah Clinch, Adrian Friday, Ivan Elhart, Thomas Kubitza, and Bholanathsingh Surajbali. Personalisation and privacy in future pervasive display networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, New York, NY, USA, 2014. ACM [DLC+14].

6.1.1 Evaluating the Integration of the Architectural Elements

Our architectural elements Yarely, Tacita, and Mercury, together with the Content Descriptor Sets used to communicate content items and scheduling constraints, form part of our overall architecture for supporting viewer appropriation in future open display networks. Whilst the elements can each be used individually (as illustrated by many of the demonstrations and deployments described in Section 5.6), this section focuses on evaluating the degree to which the elements provide integration points for connecting with other portions of the architecture.

Figure 6.1 shows the overall integration of our three architectural elements. Yarely, Tacita, and Mercury are connected through multiple integration points that supports display appropriation.

Yarely supports receipt of content schedules from Mercury and other content descriptor factories through the Subscription Management component's support for Content Descriptor Sets (CDSs). Yarely currently supports CDS subscriptions though two pull-based (polling) mechanisms – fetching files from an HTTP server, or from the local file system. Since Mercury makes its CDSs available over HTTP, these two elements can immediately be connected. Furthermore, the CDS format already provides mechanisms for describing both web content and slideshows of traditional media meaning that no extensions to the format are currently needed to export our existing set of applications. Similarly, the CDSs constraints element provides support for the ordering constraints associated with Mercury's playlists. In future, as new application and constraint types emerge, the CDS's extensible format should allow the easy addition of new elements and attributes.

A second connection between the Yarely display node and Mercury can be found in the interfaces to add deployed displays to the application store. Two such interfaces are provided: one as part of Mercury's web UI, and one within

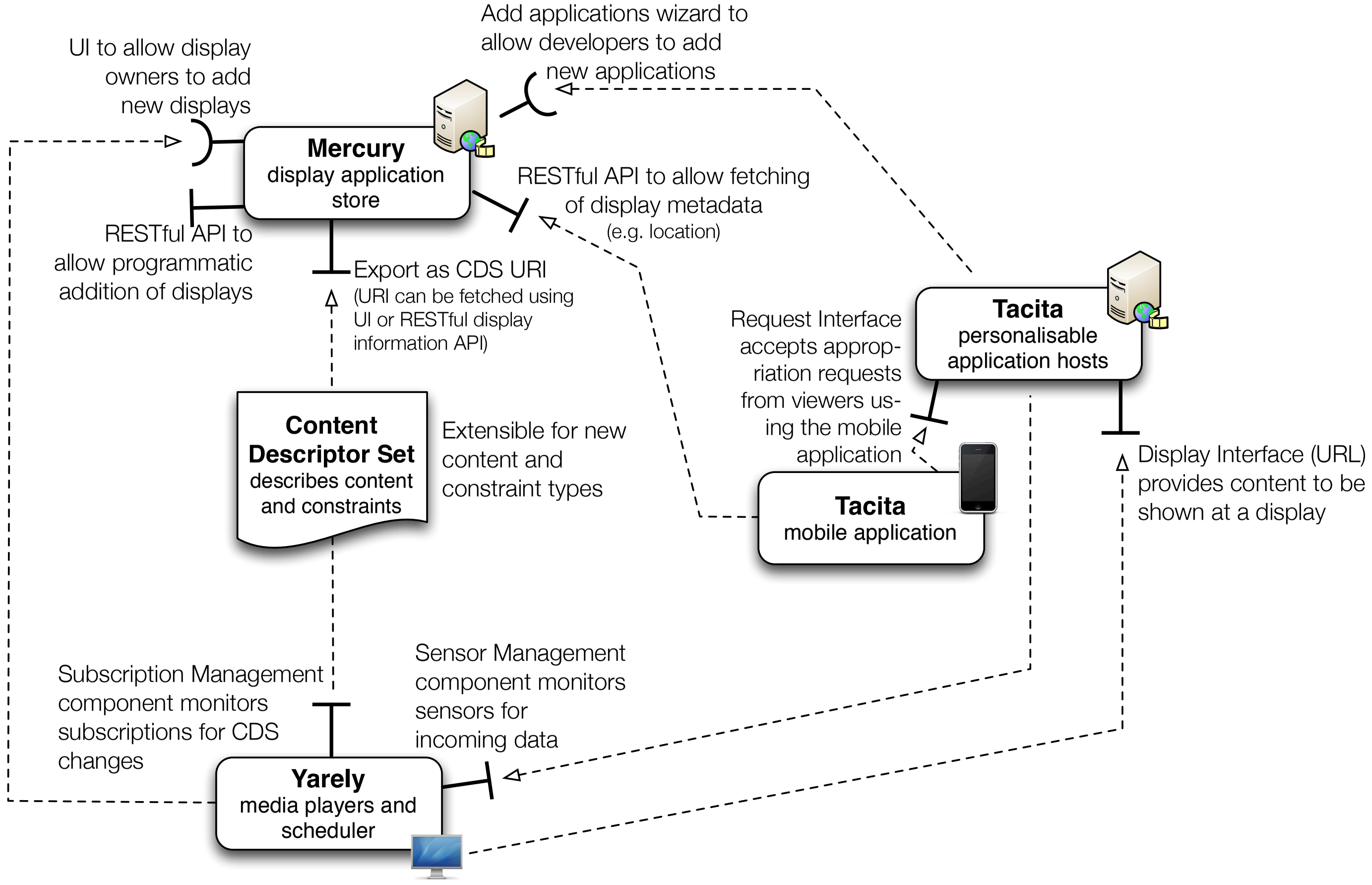

UI to allow display owners to add new displays
Mercury
display application store
Add applications wizard to allow developers to add new applications
RESTful API to allow fetching of display metadata
(e.g. location)
RESTful API to allow programmatic addition of displays
Export as CDS URI
(URI can be fetched using UI or RESTful display information API)
Tacita
personalisable application hosts
Request Interface accepts appropriation requests from viewers using the mobile application
Content Descriptor Set
describes content and constraints
Extensible for new content and constraint types
Tacita
mobile application
Display Interface (URL) provides content to be shown at a display
Subscription Management component monitors subscriptions for CDS changes
Sensor Management component monitors sensors for incoming data
Yarely
media players and scheduler

Figure 6.1: Integration points for our architectural elements.

the RESTful API. Currently, displays are typically added through the web UI but future work could see the this process become part of the initial Yarely configuration phase, with the display automatically connecting to Mercury and using the RESTful API to add itself to the user's list of deployed displays.

Yarely provides a connection point for Tacita and other contextual data sources through its Sensor Management component. The Sensor Management monitors a set of physical or virtual sensors for data. Our Yarely implementation provides handlers for two such virtual sensors: one that accepts data over a TCP socket, and that accepts incoming data from HTTP clients. Tacita's display applications can post data to the virtual sensors in order to request immediate scheduling to the display.

Tacita's display applications also provide integration points for Yarely, providing them with a URL from which they can fetch content for showing at a display. These integration points are found in each application's Display Interfaces. Opening an HTTP request to the Display Interface URL results in the application generating the appropriate content and returning the layout as an HTTP response. Since Yarely's renderer set includes web rendering on all three platforms, these components can easily be connected.

Tacita also connects to the application store through two interfaces provided by Mercury. Firstly, Tacita's display applications can be added to the application store as content items. For example, the World Clock application presented as part of Tacita in Section 5.5.2.4 was then deployed to the application store as part of our initial application set (as described in Section 6.3.2.3). The process of adding an application is completed using the Mercury web UI. Secondly, Tacita can use Mercurys RESTful API to fetch information about displays, their locations, and the applications that are available on those displays. This allows the Mercury to replace Tacita's optional Map Provider component.

Overall, it's clear that whilst they do have value as independent systems, our three architectural elements can be tightly integrated to provide an overall display architecture to support user appropriation.

6.2 Quantitative Evaluation

In this section we provide a set of quantitative measurements that report on performance aspects of each of the three software systems: Yarely, Tacita, and Mercury. In Section 6.2.1, we begin with an exploration of Yarely's rendering performance, followed by a study of Yarely's caching behaviour and its impact on resilience to network diruptions. In Section 6.2.2, we then consider the scalability of Mercury by measuring the duration of a varied set of API calls with different numbers of applications in the database. Finally, in Section 6.2.3, we consider the timeliness of making a personalised content request using Tacita, noting the effectiveness of the different proximity detection technologies for achieving good levels of both content exposure (i.e. ensuring the requesting viewer sees the content) and accuracy (i.e. ensuring the requested content is not shown whilst the viewer is not present).

6.2.1 Yarely

In March 2012, Yarely replaced e-Campus as the dominant signage platform in use at the Lancaster University campus. Connected to the e-Channels system, the software was required to play a variety of files (predominantly images, but also video, web and streams) on the existing Mac hardware and Mac OS X operating system.

The following sections report performance measures for the Yarely deployment.

6.2.1.1 Rendering Performance

To evaluate Yarely's suitability for playing common media types, we profiled the behaviour of our Mac OS X renderers. We generated a set of files that reflected typical content items scheduled to the display by users of the system: images, video and web pages. The image files used were based on two source images from Wikimedia Commons, one greyscale and one colour; two file formats, each in four different sizes were created from each original source image (sixteen files in total). Similarly, a single open source video was encoded in two video formats and three sizes and three representative websites of different sizes were selected (Table 6.1)[1]. Finally, we also generated two PHP web pages that scaled and centred colour JPEG images from our Wikimedia image set – the use of web pages to centre images was a common technique in e-Campus and so provided a useful comparison point.

We rendered each file twenty-five times, and from the resulting log files generated mean timings for key stages in the rendering process: `launching` the renderer application, ØMQ `registration` with the manager, `preparing` (i.e. generating a full screen window containing the content), making the window visible, and `terminating`. The resulting timings can be seen in Figure 6.2.

Combining results for all our renderer types, we find that the average total duration of our five renderer stages is 2.75 seconds (standard deviation 0.47 seconds). This time excludes the time the content is visible on the screen but includes two transition phases, one to fade the item onto the screen (included in the `makingvisible` timings) and one to fade the item off the screen (included in the `terminating` timings. Our transition phase duration is configurable and is set at 0.8 seconds for our tests and for all displays within the Lancaster campus deployment; this leaves approximately 1.15 seconds of time consumed by each renderer whilst running in the background. We find a low but statistically significant correlation between file size and total duration (Pearsons correlation coefficient = 0.22, $p = 0.0$), or between pixel count and total duration (Pearsons correlation coefficient = 0.14, $p < 0.001$). To understand these results we break the durations down by renderer type and rendering stage Figure 6.2.

Looking first at the renderer stages, we find three of stages to be relatively consistent in their durations regardless of the renderer type. ØMQ `registration` (when the launched renderer connects back to its parent process) is negligible in all cases, averaging around 0.003 seconds (standard deviations between 0.0006 and 0.0008 seconds depending on renderer type). The durations associated with

[1]The source image and video files can be found at `http://commons.wikimedia.org/wiki/File:Minnie_Thomas,_9_years_old,_showing_the_average_size_of_the_sardine_knife_as_large_as_this._Minnie_works_regularly..._-_NARA_-_523456.jpg` (greyscale image), `http://commons.wikimedia.org/wiki/File:New_Year's_Eve._City_Hall._Canvas._25_x_35_cm.jpg` (colour image) and `http://www.bigbuckbunny.org/index.php/download/` (video).

Media type	File format	Resolution	File Size	
			Colour	Greyscale
Image	JPEG File Interchange Format (`.JPEG`) JFIF standard 1.01.	320 x 227	41 KB	18 KB
		800 x 568	234 KB	103 KB
		1280 x 908	497 KB	250 KB
		3000 x 2129	2.1 MB	1.7 MB
	Portable Network Graphics (`.PNG`) 8-bit colour/grayscale, non-interlaced.	320 x 227	156 KB	46 KB
		800 x 568	899 KB	307 KB
		1280 x 908	2.0 MB	727 KB
		3000 x 2129	7.2 MB	3.6 MB
Video	MPEG-4 (`.AVI`) msmpeg4v2 with stereo mp3 audio	480p (854 x 480)	156.5 MB	
		720p (1280 x 720)	284.4 MB	
		1080p (1920 x 1080)	714.7 MB	
	Theora (`.OGG`) theora with stereo vorbis audio.	480p (854 x 480)	166.8 MB	
		720p (1280 x 720)	196.9 MB	
		1080p (1920 x 1080)	908.9 MB	
Web	HTML `http://motd.lancs.ac.uk`	N/A	~4 KB	
	HTML `http://www.bbc.co.uk/news`	N/A	~1002 KB	
	HTML `http://www.lancs.ac.uk`	N/A	~1.3 MB	
	Colour JPEG images from Row 1 embedded in a web page, scaled and centred using an IMG tag.	N/A	40.3 KB	
		N/A	486 KB	

Table 6.1: Overview of media files used to profile Yarely's rendering behaviour. The JPEG image format is the dominant format for files contributed to e-Channels (and therefore also dominates as the main media format within our deployment); for comparison we also include PNG as an alternative format. All video and web files are in colour, both colour and greyscale images are used. Web page sizes are based on network transfer size reported by Google Chrome.

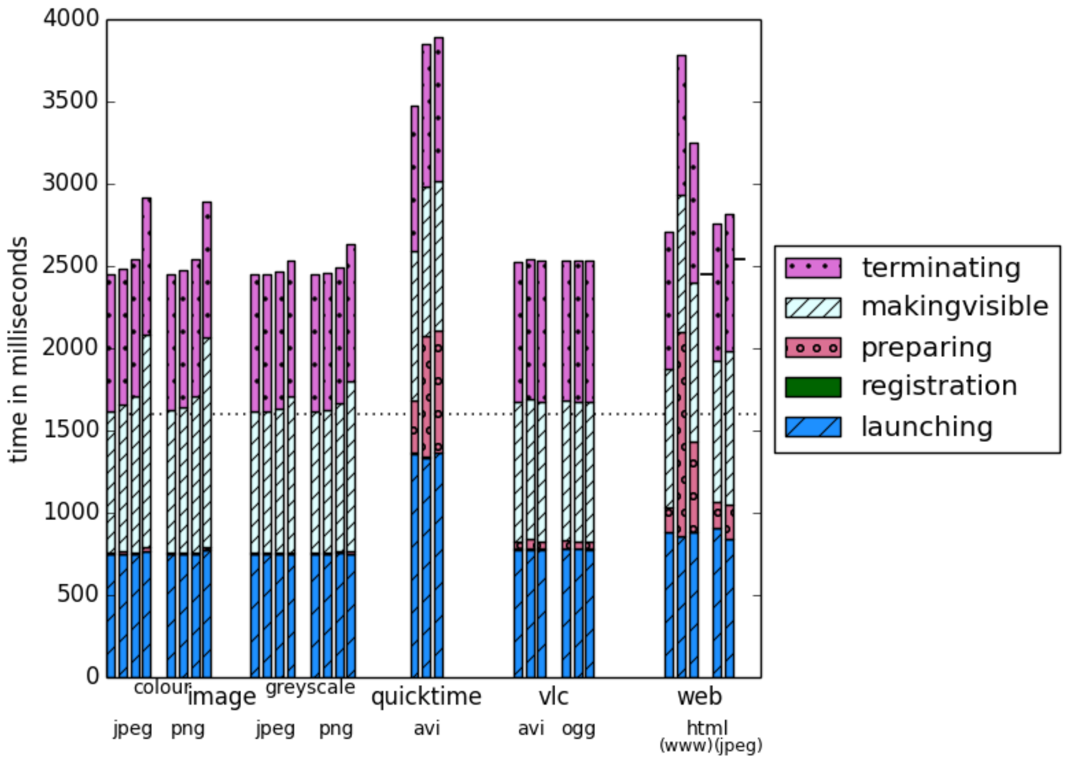


Figure 6.2: Yarely: Profiling the renderers most commonly used in our deployment: the image, video (quicktime and vlc), and web, Mac OS X renderers. Mean durations in milliseconds for each of the key stages in the rendering process (launching, registering, content preparation, making the window visible and termination) are grouped by renderer type and file format and plotted in size order: smallest to largest, left to right. Details of file formats and sizes is given in Table 6.1.
Note that the notches on the two rightmost bars represent the times taken to plot the same content with the image renderer. The dotted line plotted at 1,600 milliseconds represents a baseline value of the combined minimum `making visible` and `termination` times based on the fade-in and fade-out effects used to transition content on/off the screen (configured at 800 milliseconds each).

the makingvisible and terminating stages are clearly dominated by the 0.8 second fading effects: mean durations for the makingvisible stage lie between 0.85 and 0.95 seconds (for the VLC and image renderers respectively), termination durations are slightly lower with mean values between 0.83 and 0.87 seconds (for the image and quicktime renderers respectively). The remaining two renderer stages, launching and preparing vary with renderer type and will be explored in the the following paragraphs.

Our image renderer shows the strongest correlation between file size and duration with a Pearsons $r = 0.96$ ($p = 0.0$) for greyscale images and $r = 0.74$ ($p = 0.0$) for colour (combined $r = 0.77$, $p = 0.0$). The strong effect of file size can be seen visually in the final bar of each image file type group in Figure 6.2. Between the third and final bars we see an increase in file size that is, on average, an order of magnitude greater than the previous file size increase (mean file size increase from 1280x908 to 3000x2129 is 2857.10 KB compared with 494.75 KB for the increase from 800x568 to 1280x908), this dramatic increase in file size is matched by a corresponding increase in the size of the final bars in each group. Figure 6.2 shows that file format (i.e. JPEG vs. PNG) has little effect on the durations of each stage – although the PNG file sizes are many times larger than the JPEGs to represent the same resolution, rendering times are similar. Unsurprisingly, correlation betwen pixel count and duration is also high, with a Pearsons $r = 0.87$ ($p = 0.0$) for greyscale images and $r = 0.94$ ($p = 0.0$) for colour.

Yarely provides two renderers for showing video files on Mac OS display nodes: the default video renderer uses the QTKit to interface with the native media player, and a VLC renderer that uses the libVLC media framework to provide support for a wider range of video types and media streams. Yarely uses the default (quicktime) renderer for all supported file types, using the VLC renderer only for videos and streams not supported by the default renderer. However, our results show that the VLC renderer performs considerably faster than the native quicktime renderer, with the combined duration typically reduced by one third for each of our sample MPEG-4 files, suggesting that use of the VNC renderer for all supported files may well be preferable. Breaking down this difference further, we note that the quicktime renderer has a considerably higher launching duration than the other renderers (mean 1.35 seconds vs. means of 0.75–0.87 seconds for other renderer types). Similarly, the quicktime renderer takes over ten times as long in the `preparing` phase as the VLC renderer (mean 0.60 seconds compared to VLC's 0.05 seconds) and is also more variable (standard deviation 0.28 seconds compared to VLC's 0.03 seconds). The VLC renderer performs much more consistently overall, with a mean total duration of between 2524.04 and 2540.40 seconds across both media types and all file sizes (overall standard deviation is 0.03 seconds). Like the image renderer, the VLC renderer shows no visible effect of file format on duration. Due to its tight performance across all files, the VLC renderer shows no statistically significant correlation between file size and total duration, nor any correlation between file size and total duration (both relationships have a Pearsons $r = 0.04$, $p > 0.5$). For comparison, the Pearsons correlation coefficient values for the quicktime renderer are considerably higher at $r = 0.50$ (p<0.00001) for file size and duration, and $r = 0.53$ (p<0.00001) for pixel count and duration.

The web renderer exhibits slightly higher `launching` durations (~0.1 seconds longer) than the image and VLC renderers (but still considerably lower than the quicktime renderer). However, like the QuickTime renderer, the web renderer demonstrates very much higher preparation times (means 0.15 seconds for motd, 0.54 seconds for the lancaster home page, 1.24 seconds for BBC news, and 0.15 and 0.21 seconds for our 41 KB and 497 KB images respectively) and very high variability in those times (standard deviations 0.18, 0.45, 0.31 seconds for motd, Lancaster, and BBC news respectively, 0.20 and 0.31 seconds for our 41 KB and 497 KB images). The web renderer also shows a statistically significant correlation between file size and duration $r = 0.57$ ($p = 0.0$).

The web renderer also provides a demonstration of how the rendering provision in Yarely provides improvements on the previously deployed e-Campus system. In e-Campus images were rendered using a WebKit based renderer; each image was embedded in an HTML web page, using PHP to scale and centre the image. By contrast, Yarely uses a dedicated image renderer. Yarely's image renderer provides support for a wider range of files than could be support by the HTML `img` element. In addition, our performance statistics show that rendering a JPEG of approximately 40 KB or 500 KB takes around 0.3 seconds longer (and has greater variability) with the web renderer than with the image renderer.

6.2.1.2 Resilience

Yarely uses local storage of content files and content descriptor sets (CDSs) in order to increase resilience (particularly in times of network failure). Media files associated with content items are cached prior to the first playback – when the item is due to be scheduled again, the file cache is checked for matching entries to prevent items from being repeatedly fetched over the network. If the network becomes unavailable, the scheduler will detect that uncached items cannot currently be played and will temporarily drop them from its playlists, restricting playback only to those items it has available in its file cache. Similarly, the content descriptor sets themselves are stored locally in a SQLite database. Unlike the media cache, the subscription manager will continue to monitor the original source for updates. However, if the original source becomes unavailable then Yarely's subscription manager will continue to make the persistent copy available to other components.

Since our Lancaster deployment of Yarely sources the majority of its content and CDSs from e-Channels (this will be discussed further in Section 6.3.2.2), we use Apache web server logs from e-Channels in order to study the effect of Yarely's media caching and CDS persistence strategies. We use a snapshot of logs captured between 1st April 2012 and 11th February 2013. During the study period, the median number of displays using Yarely and e-Channels is eighteen per hour (seventeen on weekends and nineteen on weekdays).

The difference in Yarely's media caching and CDS persistence strategies can be seen in Figure 6.3. Plotting requests for the display CDSs against requests for content media shows that content is cached effectively (resulting in a low number of requests) whilst CDSs continue to be monitored for changes (resulting in a larger number of requests). The median number of content items fetched per hour is zero (as are both the 25% and 75% quartiles) but the maximum is 545 (most likely representing the point at which the Yarely has started with an empty cache). The number of CDS requests has an approximately linear relationship with the number of active displays. The median number of CDS requests per hour is 511 (IQR 509).

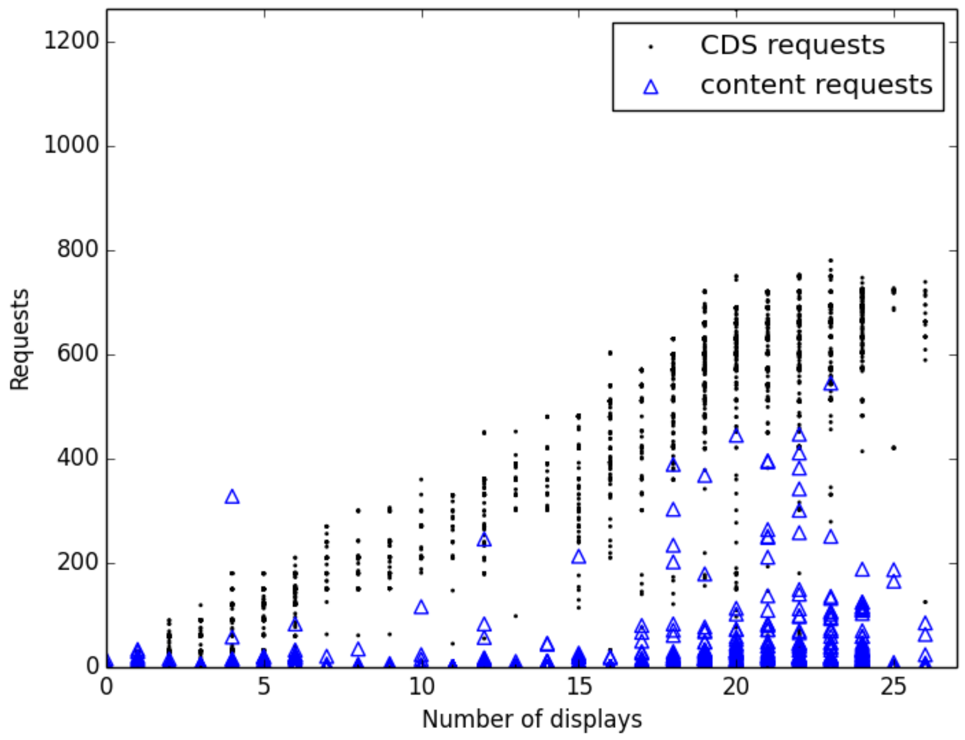


Figure 6.3: Yarely: Reducing content retrievals through file caching. The number of Content Descriptor Set (CDS) requests is approximately linear with number of active displays, whilst the number of content items fetched is lower due to the effective caching strategy (based on a snapshot of log data gathered between 1st April 2012 and 11th February 2013).

In addition to the above statistical analysis, our experiences from the Yarely demonstrations and deployments have also contributed to our understanding of how CDS persistence and content caching improve Yarely's resilience and reliability. During demonstrations at PerDis 2012, an extended period of network unavailability early in the conference posed a challenge for a number of systems. Prior to the network outage Yarely had successfully cached its CDSs and a small portion of its allocated content (~3 items) – the scheduler cycled through these items until the network was reinstated. Similarly, one display in our campus deployment was inadvertently disconnected from the network. After some time the owners reported that the display didn't appear to be updating with the most recent content and only then was the network disconnection discovered, the node had continued to operate despite being disconnected (and continued to operate in this way for a further two weeks whilst an additional network point was installed).

6.2.2 Mercury

In this section we present a brief consideration of the scalability of Mercury. At present Mercury is deployed on a single virtual machine with modest resource allocation. If usage of the store increased, a number of improvements for scalability and performance could be made, for example by increasing resource provision, or introducing caching. For this reason we provide only a limited evaluation of the current scalability and performance of the Mercury deployment.

The results presented in this section were gathered using an automated test script running API calls against the live deployment. Each API call was called with a varying number of applications already in the App Store. Each application was created with a single billing model. One user review was added for each application. The overall timing of the API call (a network operation) was recorded. Each measurement was repeat to produce a total of thirty measurements per combination of API call and number of applications.

Figure 6.4 shows that the time taken to add a new application does not appear to be impacted by the quantity of applications already in the store. The duration of the call is moderately variable overall, most likely due to variabilities in network speed, but the number of applications itself has a negligible effect on the length of the call. For example, adding an application with between 0–4,999 pre-existing applications in the database takes on average 44.39 milliseconds (median, with an IQR of 20.39 milliseconds), similar timings can be seen when adding an application with 5,000–9,999 pre-existing applications in the database (median 44.48 milliseconds, IQR 20.42) and with 10,000–14,999 pre-existing applications in the database (median 44.54, IQR 20.41).

Similarly, the call to purchase an application remains constant regardless of the number of applications in the database (Figure 6.4). The call takes a median of 29.17, 29.15, and 29.20 milliseconds with 0–4,999, 5,000–9,999, and 10,000–

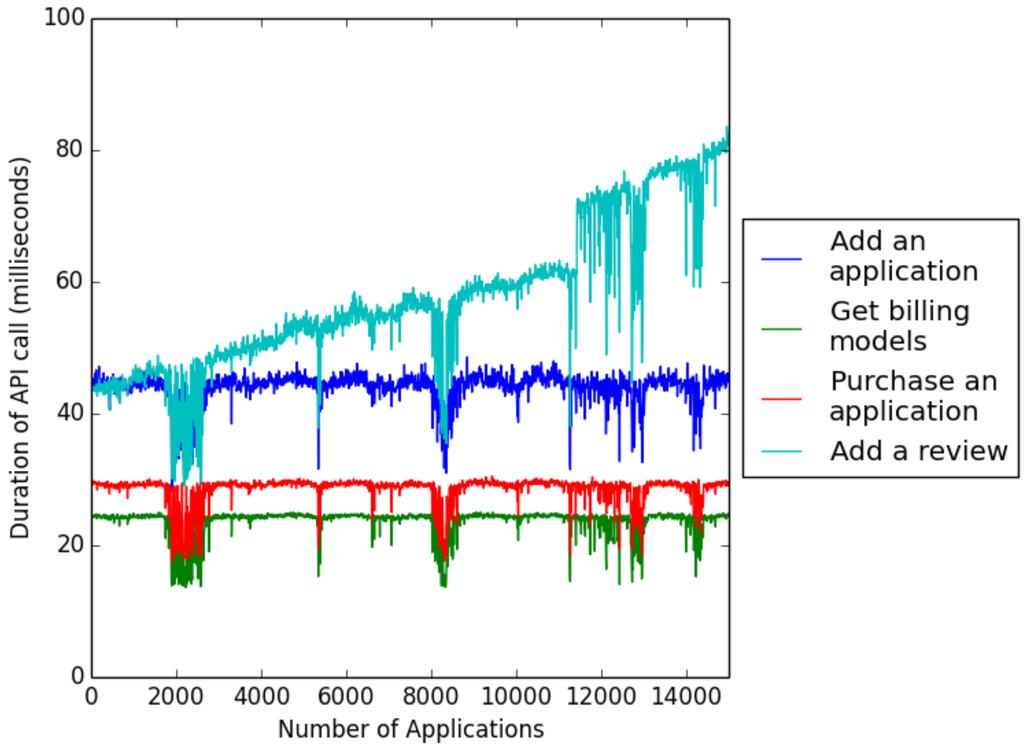


Figure 6.4: Mercury: Median duration of the add application, get billing models, purchase application, and add review API calls with varying numbers of applications in the database (0–15,000).

14,999 applications in the database respectively (interquartile ranges 13.31, 13.30, and 13.28). Getting the billing models associated with an application also appears to be largely constant in duration (Figure 6.4) taking a median of 24.33, 24.34, and 24.31 milliseconds with 0–4,999, 5,000–9,999, and 10,000–14,999 applications in the database respectively (interquartile ranges 11.50, 11.53, and 11.47).

The final call shown in Figure 6.4, represents the process of adding a review. This call clearly is impacted by the number of applications stored in the database. For example, adding a review with 0–4,999 applications in the database takes on average 46.45 milliseconds (median, with an IQR of 22.73 milliseconds), with 5,000–9,999 applications in the database the duration of the call is increased by almost ten milliseconds (median 55.64, IQR 27.05), and with 10,000–14,999 applications the duration is increased by a further eight milliseconds (median 63.51, IQR 35.67). No corresponding increase is seen in the size of the network data transferred (median result size 832.0 bytes and IQR of 256.0 for 0–4,999 and 5,000–9,999 applications, and median 835.0 and IQR of 254.0 for 10,000-14,999 applications). We attribute the difference in time taken to add a review to a more complex relational structure in this case (this is explored further on page 255).

Comparing all four calls shown in Figure 6.4, we observe timely responses of under 0.1 seconds.

To explore the performance of application retrieval operations, we measured three different methods of searching for applications: search by name, search by description, and search by review. Each search was conducted for a range of expected response ratios: 1 application, all applications, 50%, ~30%, and 10% of applications. We also measured two list operations – one to list all applications, and one to list a single application identified by application ID.

Figure 6.5 shows the median duration of each of our three kinds of search (averaged across all result ratios). We see that all three searches exhibit some increase as the number of applications grows but that this effect is significantly greater when searching by review text. Searching by name with 0–499 applications in the database takes an average of 31.16 milliseconds (median, with an IQR of 145.49 milliseconds), this same search takes a median of 135.88 milliseconds (IQR is 178.48 milliseconds) with 500–999 applications in the database and 135.45 milliseconds (IQR is 180.56 milliseconds) with 1,000–1,499 applications in the database. Similarly searching by description takes an average of 33.43 milliseconds (median, with an IQR of 147.87 milliseconds) with 0–499 applications in the database compared with 136.05 milliseconds (IQR is 170.92 milliseconds) with 500–999 applications and 137.87 milliseconds (IQR is 175.68 milliseconds) with 1,000–1,499 applications. By contrast, the increase in time taken when searching by review with more applications is much greater. Searching by review with 0–499 applications in the database takes an average of 34.43 milliseconds (median, with an IQR of 339.04 milliseconds), this same search takes over ten times as long with a median of 401.44 milliseconds (IQR is 776.71 milliseconds) with 500–999 applications in the database and increases by half again to 656.72 (IQR is 1110.32 with 1,000–1,499 applications in the database. The effect of increasing the number of applications upon time taken to search by description is particularly marked in Figure 6.5 because of the overall greater time taken to do this search for even low numbers of applications. Unlike the two other searches, searching applications by review text requires reference to an additional entity, the review, rather than just the application objects alone. We attribute the poorer performance of the searches in this case to the additional database table lookup.

Looking at the size of the network data transferred (Figure 6.6) we see a sizeable increase in the size of the result as the number of application in the store grows, but only for approximately the first twenty applications. Beyond twenty applications, the network traffic returned becomes relatively steady at approximately 45 KB. We attribute this pattern to the pagination provided by Mercury's RESTful API. Both search and list results are paginated at twenty results per page. Each query will result in between zero and twenty results, with pointers for the URLs to subsequent and previous pages. The resulting data sent over the network will therefore contain the JSON representation of no more than twenty applications. If we remove non-paginated data from our results (i.e. searches that return fewer than twenty results), we see a significant increase in the median time taken, but the difference between call durations for small and large numbers of applications is greatly reduced when searching by name or description. Median time taken with 0-499 applications in the database is 193.73 milliseconds (IQR 128.75) when searching by name and 216.04 milliseconds (IQR 131.01) when searching by description, compared with medians of 194.03 mil-

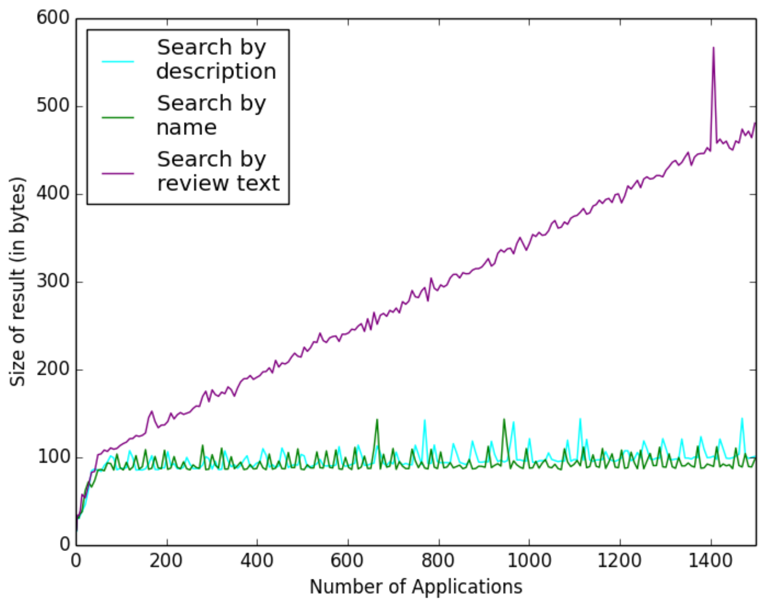


Figure 6.5: Mercury: Median duration of the search applications API call (no ordering specified) with varying numbers of applications in the database (intervals between 0-1,500). This plot shows the combined median duration for a set of searches returning a range of ratios of the complete dataset (1 application, all applications, 50%, ~30%, and 10% of applications.

liseconds (IQR 130.15) when searching by name and 190.76 milliseconds (IQR 132.15) when searching by description with 500–999 applications in the database, and medians of 196.83 milliseconds (IQR 131.38) when searching by name and 199.18 milliseconds (IQR 133.49) when searching by description with 1,000-1,499 applications in the database. Searching by review continues to grow more expensive (and more variable) as the number of applications increases: median time taken with 0-499 applications in the database is 393.33 milliseconds (IQR 206.89) compared with 792.74 milliseconds (IQR 403.61) with 500–999 applications in the database and 1127.47 milliseconds (IQR 677.34) with 1,000-1,499 applications in the database.

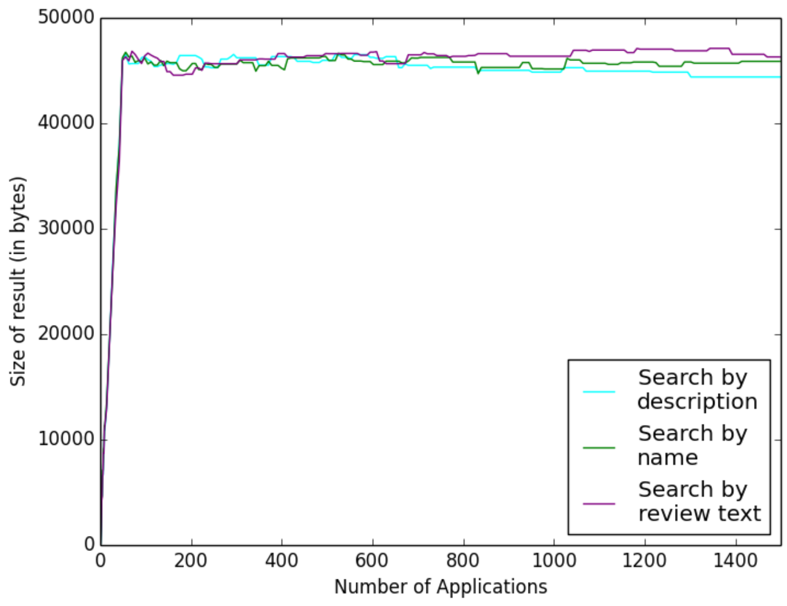


Figure 6.6: Mercury: Median size of the result data from the search applications API call (no ordering specified) with varying numbers of applications in the database (intervals between 0-1,500). This plot shows the combined median duration for a set of searches returning a range of ratios of the complete dataset (1 application, all applications, 50%, ~30%, and 10% of applications.

Figure 6.7 shows the performance of the three searches when compared to the list operations – Figure 6.7a shows the use of listing and searching to fetch all applications in the store, and Figure 6.7b the use of listing and searching to fetch a single application. We see that returning a single application is typically very fast, with a median duration of less than 30 milliseconds in all cases (Figure 6.7b). Returning all applications is more expensive, and we again see the cost of querying an additional table when searching by review – whilst the other retrieval operations show no increase in the median duration or variability between our three database size bins (20–449 applications, 500–999 applications and 1,000–1,499 applications in the database), the median duration for searches by review increases from 475.80 milliseconds (IQR 267.37) to 1140.39 milliseconds (IQR 341.60) and then 1786.76 milliseconds (IQR 358.04).

Finally, Figure 6.8 shows the effect of adding an ordering to each of our three searches. We note that due to a default ordering clause on our Django views, searches with no order specified are in fact ordered on three keys. As a result of this default ordering we see no noticeable difference between the performance of searches with or without ordering.

6.2.3 Tacita

The appropriation models described in Chapter 3 (walk-by, active, and longitudinal) require a variety of levels of timeliness for proximity detection. In the most demanding use case of walk-by personalisation there is only a small time window in which it makes sense to display information directed to a specific user.

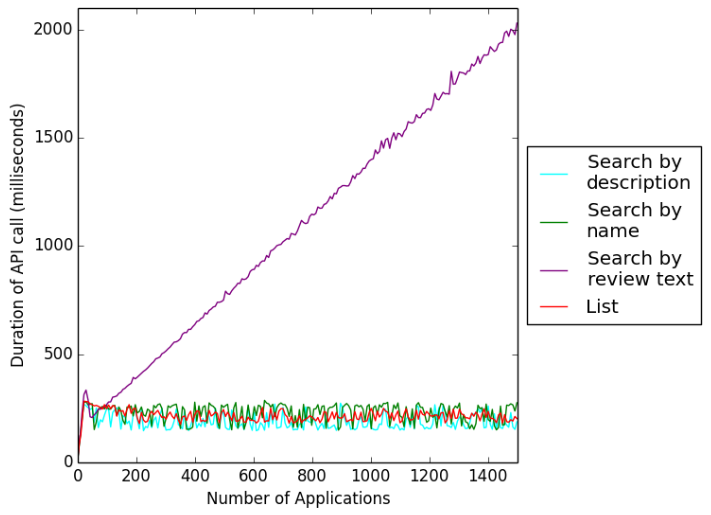
The elapsed time from the moment a Tacita mobile client enters a display's triggering area to the moment the display changes can be divided into three components.

1. The time it takes a mobile client to detect that it is within a display's triggering range and to determine which applications to contact.

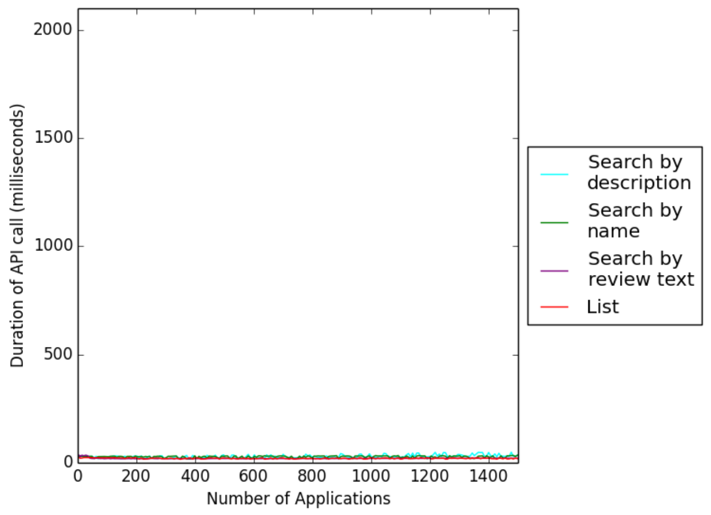
2. The time taken for a mobile device to make a request to an application though a GPRS, 3G, 4G or WiFi connection.

3. The time taken by an application to recognise the user request and to initiate a corresponding change at the public display.

Measurements for these three components were conducted by a team that included the author and are reported in Davies et al. [DLC+14]. They are as

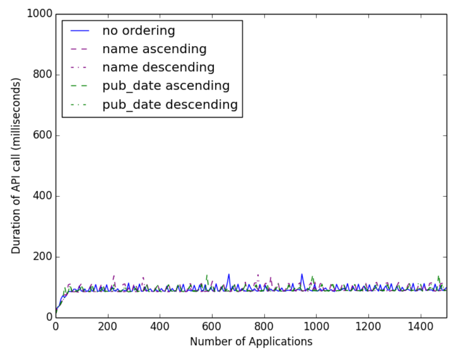


(a) Return all applications

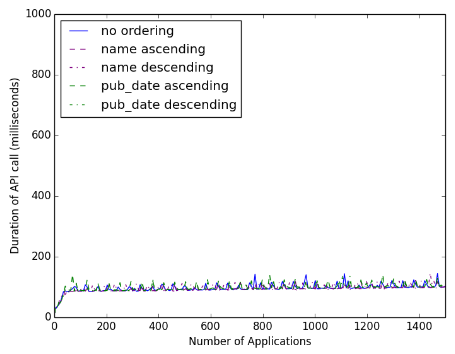


(b) Return one application

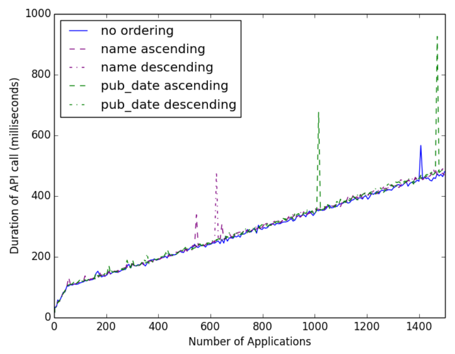
Figure 6.7: Mercury: Median duration of the list and search applications API calls (no ordering specified) with varying numbers of applications in the database (intervals between 0-1,500).



(a) Search by name



(b) Search by description



(c) Search review text

Figure 6.8: Mercury: Median duration of the search applications API calls with various ordering parameters for varying numbers of applications in the database (intervals between 0-1,500).

follows: component one varies with mobile phone sampling rates for the different location technologies (typically around 1–5 seconds when activated); component two averages 1.96 seconds (±0.99 seconds) using 3G communication and the Android mobile client but may be faster or slower depending on the technology used; and component three averages at 2.35 seconds (±0.19 seconds) using a display testbed in which displays poll for schedule changes at a rate of 3 seconds.

To confirm the appropriateness of the Tacita for providing timely personalised display updates, the research team also conducted a series of walking trials in two real-world environments [DLC+14]. Using the Tacita Android mobile client and a web-based signage player, values were calculated for *content exposure* and *content accuracy*. Content exposure is the percentage of time that the content was shown on the display while the viewer was within range and characterises the effectiveness of the system at showing content to the viewer. Content accuracy is the percentage of time that the content could have been seen by the viewer relative to the total amount of time that the content was shown. For example, a system that continuously showed a single content item for a target user irrespective of whether that user was present would have a high content exposure but low accuracy.

This work demonstrated that given an appropriately sized trigger zone, the system was able to show personalised content for at least some of the time that the viewer was in the viewing area: exposure levels for Bluetooth were 21–58%, GPS 25–64%, WiFi 53–83%, and combining technologies raised the exposure levels to 89-98% [DLC+14]. Reasonable accuracy was also achieved – in the best cases an average of 65% of the time that the content was on the screen the intended viewer was also within the viewing area of the screen.

The evaluation of Tacita summarised above used a web-based signage system that used polling as a mechanism for detecting updates to the content schedule. Integrating Tacita with Yarely has the potential to offer improvements in performance as Yarely's Sensor Management component allows event-based triggering rather than relying on polling intervals.

We conducted fifty measurements of our Sensor Management component using the socket sensor to receive a web request (i.e. a URL to be shown). We found the mean duration from initial socket connection to scheduling of the received item to be 204.02 milliseconds (standard deviation 0.62 milliseconds). The majority of this time is consumed by the socket sensor as it accepts the data over the socket (mean 201.39 milliseconds, standard deviation 0.49 milliseconds, from TCP connection to receipt of the final data packet). Processing and ØMQ communication within Yarely's Sensor Management component accounts for 1.69 milliseconds (mean, standard deviation is 0.58 milliseconds). The remaining 0.94 milliseconds (mean, standard deviation is 0.42 milliseconds) are taken in communicating the sensor data to the Playlist Generation and Scheduling components and in the time taken to break out of the main scheduling loop and switch to viewer content.

Combining this data with the rendering performance measurements from Section 6.2.1.1 we estimate that the time taken from receipt of a TCP connection for the display appropriation request to the appearance of the requested item at the display is approximately 2427.02 milliseconds (i.e. 2.4 seconds), this includes a 0.8 second transition in which the content is faded onto the display. This is inline with the Tacita performance reported in Davies et al. [DLC+14] and suggests that our overall architecture is able to support effective display appropriation using viewers' personal mobile devices as a source for requests.

6.3 Qualitative Evaluation

6.3.1 Stakeholder Attitudes to Appropriation

6.3.1.1 Display Owner Evaluation

Establishing that display owners will accept appropriation requests from viewers is a critical evaluation point. We interviewed six display owners: three e-Campus users, two who controlled displays running a commercial digital signage software, and one user who owned both an e-Campus display and a display that showed a continuous Powerpoint presentation for digital signage purposes. The modal number of displays owned by our display owners was one or two, the maximum was seven. Displays managed by the owners included a mixture of screens in academic space, residential space, social space, and public commercial areas such as the theatre and sports centre. Five of the owners reported a strong sense of ownership over the displays.

Display owners were asked some general questions about their display and about the role of appropriation. We also asked a series of questions in which answers were arranged along a continuous or categorical scale (see Figure 6.9). The owners were asked to shade the scales indicating the portions that were acceptable to them for their display, heaviest shading indicated that they were completely happy with this portion of the scale (i.e. a likert scale '5'), and no shading indicated that they would not accept this for their displays. It was hoped that this method would allow us to enquire about multiple approaches to a given topic in a concise manner[1] and would provide a clear graphical reference for further discussion with each owner. The questionnaire results were coded independently by two researchers who each identified changes in shading and allocated each distinct shaded portion a value from zero to five. Where the final coded responses differed (no answers differed by more than one scale point), an average of the two codes was taken.

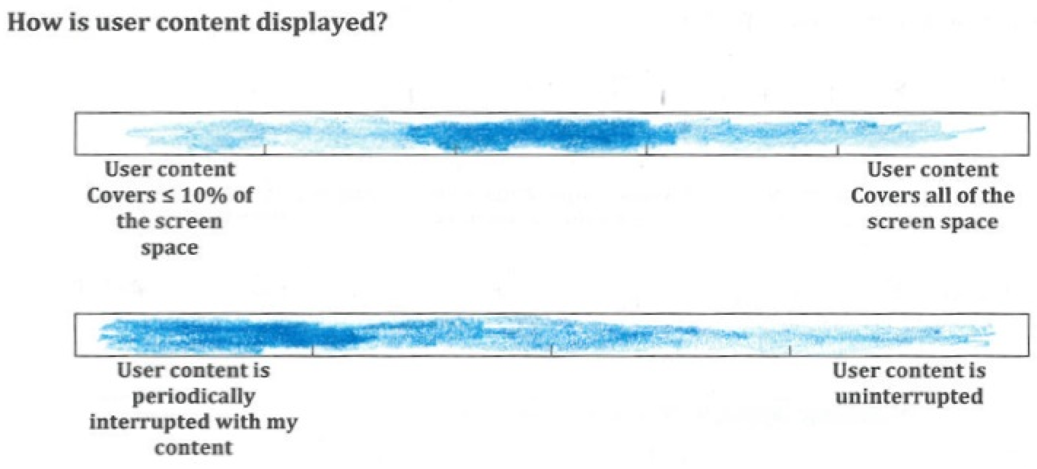

How is user content displayed?
User content Covers ≤ 10% of the screen space
User content Covers all of the screen space
User content is periodically interrupted with my content
User content is uninterrupted

Figure 6.9: Sample Questionnaire Answers

When asked about the introduction of user appropriation to their screens, our display owners were most positive about accepting appropriation from people belonging to a specific, known group (median coded rating 4.5) or those known to

[1]For example, the questions represented in Figure 6.9 could also have been represented with a much longer series of questions asking about specific points on the scales – 10% user content vs. 25% user content vs. 50% user content etc.

them personally (median coded likert rating 4.0). For example, discussion during questionnaire completion showed that the display owners were typically positive about allowing appropriation requests from students, sport centre members etc.. Appropriation requests from unknown viewers were neutrally rated (median rating 2.5) as long as some mechanism was provided for recording identity; coded ratings reduced to 1.0 for appropriation requests from viewers whose identity was not recorded.

Our display owners had varied approaches and concerns when it came to moderation of user content. Moderating content ahead of time introduces a burden to those doing the moderating, particularly as the quantity of user content increases – half of our owners indicated that moderation ahead of time would not be acceptable to them with many citing practicalities as a key motivator. Unfortunately, alternative strategies introduce an element of risk. For example, one might block content based on abuse reports or retrospective moderation, in this case an item of inappropriate content might show on the screen but subsequent requests for that item (or requests by that user) would not be permitted. Once again, half of our owners did not consider this method acceptable (but overall this was the most positively regarded method with a median coded rating of 2.5). Only one of our display owners considered "no moderation" as a possible option for their display and rated this considerably less strongly than both of the other options.

Restricting user appropriation to a predefined set of applications is one method by which display owners can retain control over display content. Applications also represent a stark contrast to most current public display content. When asked about the introduction of user applications, our display owners were largely positive in their responses as long as the application itself was known to the display owner (median rating 4.5). Encouragingly, allowing viewers to customise their selected application using parameters (as supported in Tacita) made little impact on acceptability to display owners (coded median just 0.25 lower). Opening up the display to "any content requested by the user" (as in our VNC application) was generally considered to be completely unacceptable (median rating 1.0).

Our display owners were quick to see the benefits of display appropriation for passing viewers, particularly in the context of the wider network. All but one owner indicated agreement or strong agreement with the statement "User appropriation would improve the University campus displays" (the remaining owner gave a neutral response). Owners saw slightly less value in enabling appropriation on their particular display(s), typically providing a neutral response. During the interview some of the owners justified this by commenting that they felt the content on their display was already highly targeted for the location. Overall only one display owner was particularly negative towards the idea of user appropriation choosing not to shade the majority of the scales.

Consideration for the user also affected display owners answers to questions about the time and space they would give over to a user: shading was either missing or lighter between "User content covers $\leq 10\%$ of the screen space" and the midpoint (with the exception of one user for whom the midpoint was darkest, fading away on each side [Figure 6.9]); four of the six owners were happy for user content to completely cover the screen space and they typically commented that they felt this would be the most useful to those trying to appropriate the screen. Likewise, owners were typically unlikely to impose time restrictions during day to day operation - strongest agreement was for user appropriation "usually anytime but I can choose to turn it off for a period" (median coded rating 4.13). One owner was happy to have the display available at anytime without turning it off for open days, special events etc. and one indicated that they would restrict appropriation to specified time slots. Within this availability, the display owners generally (unsurprisingly) wanted to avoid having their content starved out by user content and typically suggested a balance of around 50/50 as the ideal.

When asked which benefits would motivate them to allow users to appropriate the display, owners most commonly felt that users would "feel more positive about me /my organisation" (one neutral, one agree, the remainder strongly agree). Other highly-rated benefits included "Users attend to more of my content" and "Users feel more positive about the space in which my display(s) are situated"[1].By contrast, there was little expectation that user would reimburse the display owner financially (one disagree, the remainder strongly disagree). This may be a product of the University environment in which the displays are lo-

[1]The idea that digital displays can positively impact space is supported by existing research [CMB08, DNM+10].

cated, but it is worth noting that one of our strongly disagreeing display owners represented the marketing department of the campus theatre whose patrons are members of the general public.

6.3.1.2 Validating Usage Models and applications

Our study of the Bluetooth device name system in Section 3.3.1 (page 105) indicated that a key obstacle to the success of display personalisation is the development of compelling applications for viewers. We therefore conducted two user engagement studies that focussed on exploring possible applications and the influences on application choices.

Our two studies focussed on the VNC applications and asked respondents to assume applications could be placed on a virtual desktop that could then be shown on nearby public displays. In the first study we conducted an online survey disseminated via departmental email lists (primarily technically-oriented subscribers) and through our social networks (Facebook, Twitter, etc.); some additional distribution was achieved by those initially receiving the survey choosing to share it with others. In the second study we held two small focus group sessions with participants recruited from within the University computer science department. We received $n = 68$ survey responses in our first study with some bias towards young, male respondents.[1] Our second study involved seven participants including research staff, undergraduate and postgraduate students.

Online survey

Respondents were asked to report the likelihood that they would wish to appropriate displays with a set of predetermined applications – Facebook, Twitter, Flickr, news, email, gaming, clock, map [Figure 6.10a]. Our application choices were intended to reflect current applications used on either mobile devices or existing display deployments (including a mixture of applications that had been implemented for Tacita and those not currently included in the available application set); we restricted the set to well-known applications whose functionality was easily conveyed to the user. Our results show that utility applications (clock,

[1] Our sample was 75% male. 31% of respondents identified themselves as being 19-25 years, 44% 26-35 years, 16% 36-45 years, 6% 46-55 years and 3% aged 56+ years.

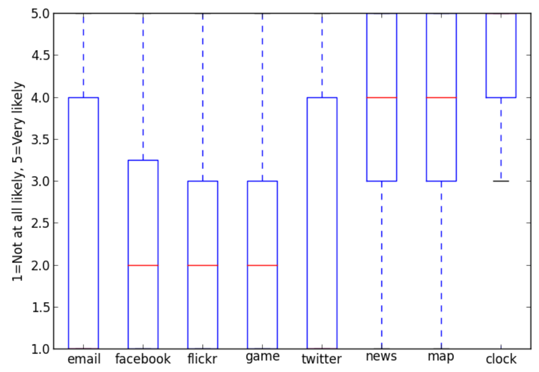
map, news) are considered to be more likely use cases (median rating 4 or 5 with low IQR) for display appropriation than social networking and other personally revealing applications (median rating 1-2, moderate-high IQR) [Figure 6.10a].

In addition to our predetermined application set, we also sought ideas for additional applications. Suggestions were typically for utility applications and included calendars/appointments (8 respondents, median likelihood of use 4.00, IQR 4.25-4.00), to-do lists (6 respondents, median likelihood of use 4.50, IQR 5.00-3.25), and weather (5 responses, median likelihood of use 5.00, IQR 5.00-4.00).

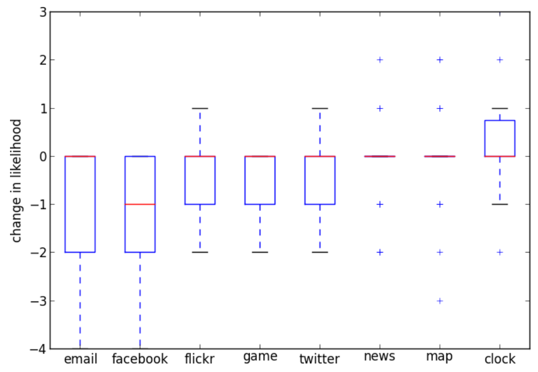
The second issue we explored with respondents was the impact of location on the utility of the proposed system. We differentiated between the level of publicness – semi-public vs. public locations – and the transitivity of a space – space in which they would typically linger vs. space that they typically 'passed through'. We also investigated if the presence of others in front of the display would influence their foraging choices. Specifically, respondents were asked if they would change their answers if they knew a senior work colleague or stranger would pass by and look at the display.

Respondents indicated that they would find the system most useful in spaces in which they expected to linger (median 1-2 scale points higher) [Figure 6.11]. This is somewhat in conflict with prior work that suggested that individuals observed waiting in a space with a display were no more likely spend time looking at the display [HKB08]. We note however that other studies that asked participants about when they expect to look at a display also found an increased likelihood for waiting areas and spaces where there was "time to watch the display" [MWE+09], so this could simply be a discrepancy between perceived and actual behaviour. Alternatively, the personalisation aspect of this study could itself account for the difference – participants may be genuinely more likely to spend time looking at a display in a space in which they would be likely to linger but only if they expect the content being shown to be of interest to them.

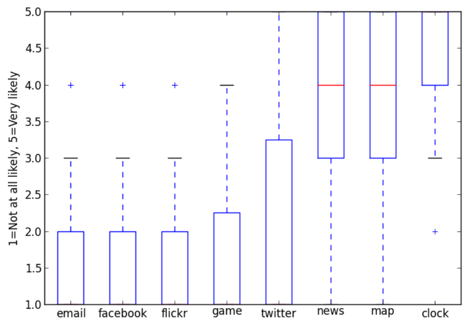
Respondents were also more positive about the utility of the system in a semi-public space than a public one (median 0-1 scale points higher) [Figure 6.11]. 50% of respondents reported that they would not alter their application choices in the presence of others – perhaps because they were already aware of the public nature



(a) Original reported likelihood



(b) Change in reported likelihood



(c) Overall reported likelihood after change

Figure 6.10: Box plots showing a) the original reported likelihood of respondents to appropriate a public display for each of the predetermined applications (where 1 indicates “not at all likely’ and 5 indicates “very likely’); b) the change in reported likelihood for those respondents who reported that the presence of a stranger or senior work colleague would change their application choices (50% of respondents): a negative number indicates that respondents become less likely to show the application, a positive number that they become more likely to show the application; and c) the reported likelihood of respondents to appropriate a public display for each of the predetermined applications once changes caused by the presences of others have been accounted for (where 1 indicates “not at all likely’ and 5 indicates “very likely’).

of the displays. Of the remaining respondents, 80% would change their application choices regardless of whether it was a stranger or a work colleague passing by (8% would change in the presence of a colleague but not a stranger, 12% in the presence of a stranger but not a colleague). When asked for their changes on a per-application basis, Figure 6.10b shows that the changes are typically very small and generally negative – that is, people become slightly less likely to appropriate a display in order to show applications. Examining the final privacy-aware likelihood for each application[1] [Figure 6.10c], we see that these small changes have the effect of polarising the existing split between utility and social-networking/communication applications: the former remain largely unchanged by respondents new-found privacy-awareness whilst the latter become even less appealing to potential users.

Respondents were explicitly told that they could use the approach as a method of sharing/broadcasting information with others and asked which applications they would be likely to share with others (cf. Figure 6.12). Again suggestions were taken for additional applications; the majority of which (8) were for items with some element of personal ownership (e.g. personal or project websites, club events), the remaining five reflected opinions they were prepared to share with others (music, movies, politics).

Focus Groups

To gain further insights into how people might appropriate displays we held two small focus group sessions. We began the sessions with an overview of the concept (again focussed on the VNC application) and underlying technologies (public displays, their forms and locations; appropriation). Where appropriate these were supported with photographs (e.g. in-situ photographs of displays in an airport, bus stop, metro, mall etc.). Once participants were familiar with the concept they were given the task of designing their own personal display layouts using paper resources.

A poster-sized (A2) paper sheet was provided as the desktop and a selection of

[1] i.e. the original likelihood of using each application for people who did not change their answers combined with the likelihood of using each application with a stranger or a work colleague passing by for those who did change their answers.

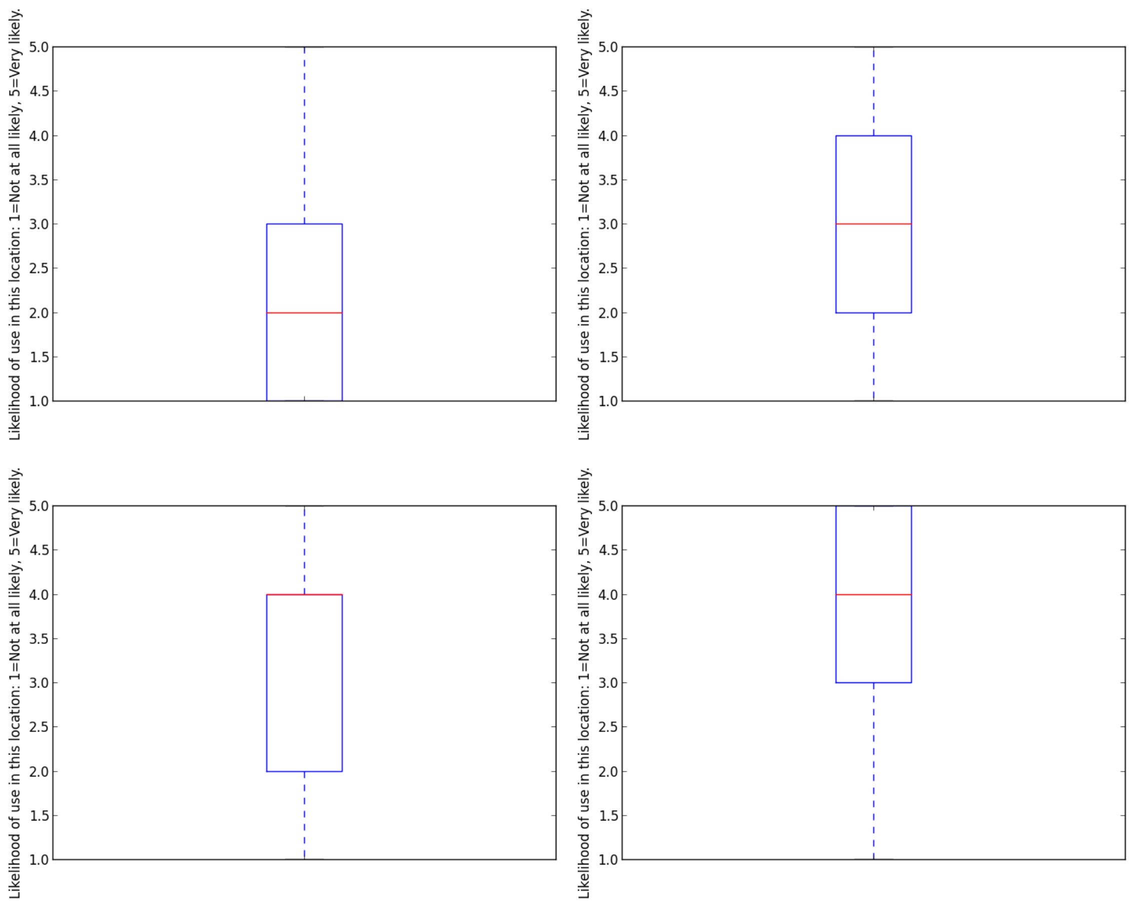


Figure 6.11: Tacita: reported usefulness of display appropriation by location type. The expectation of lingering in a space (bottom) increases the reported usefulness. Users also report a greater expectation of usefulness in a semi-public space (right) when compared with a public one (left), but to a lesser degree.

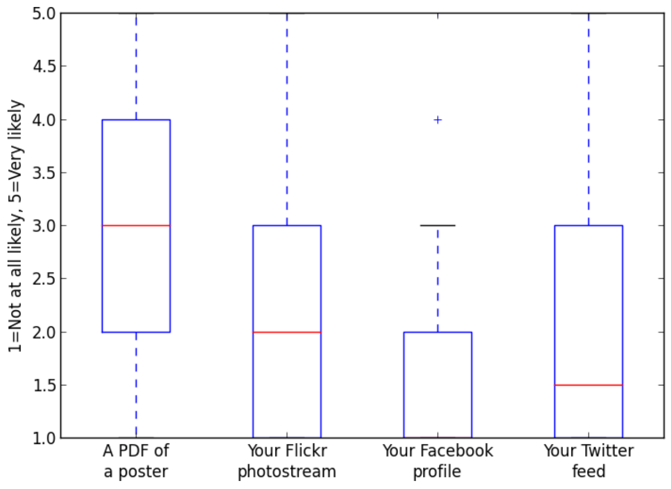


Figure 6.12: Tacita: reported likelihood of using display appropriation as a sharing/broadcasting method using a variety of applications (where 1 indicates “not at all likely”, and 5 indicates “very likely”).

printed screenshots (including popular Webpages – news, social networks, maps; common desktop widgets – clocks, calendars, weather, sticky notes; and other window applications – email, instant messaging, games) were provided as potential applications – participants were encouraged to use scrap paper and coloured pens to create additional applications if they could not find one that met their needs. Participants fed back to the group by describing the contents of their screen and the motivation for their choices. During these motivation discussions participants freely engaged with each other regarding privacy, broadcast/sharing models, location and localisation (amongst other topics). As a final discussion point, participants were asked to consider their expectations for the system if more than one user was in-range of the display at a given point in time. During the focus groups one of the researchers created a live demo of a system with the suggested content on the display. This was shown to participants following the design work and was used to stimulate further discussion on more speculative ideas.

Participants created many different screen layouts containing a wide variety of information (e.g. Figure 6.13). A number of themes emerged:

Focus on traditional display content: All of the participants included a significant amount of “traditional" signage content such as news feeds (six of our seven participants) and weather reports (5/7). However, most participants wanted to see their "own" news feeds rather than generic news information – suggesting a strong requirement for display personalisation. Reinforcing our findings from the survey, many participants created sketches that included clocks (5/7) and calendars (6/7).

Implicit localisation: All participants included sketches of information feeds that made an implicit assumption that the information could be customised based on the display's location. For example, local events listings or maps. Localisation was also a key feature in participants discussion of their display content. We did not explicitly state the availability of this behaviour yet participants seemed to assume that this would be possible.

Limited use of social media: While all participants claimed to have at least one social media account (typically Facebook), the layouts made little use of social media feeds – only three display layouts contained them (one layout contained both Facebook and Twitter, two contained one but not the other).

Simplistic models of sensitivity: Many participants had concerns about privacy, which directly influenced their choices regarding the use of social media. Participants also sketched out applications that filtered information before displaying it – for example showing an agenda but with the actual events blanked out so that a user could tell they had upcoming appointments but other viewers could not discern the nature of those appointments. However, participants in general had very simplistic models of information sensitivity and appeared unaware that many of the information feeds they were proposing might compromise their privacy, even though it did not contain explicit identifiers (e.g. a personal to-do list). Moreover, none of the participants appeared to consider privacy from the perspective of the suppliers of information (for example the *sender* of an email that a user chooses to display on the screen), nor did they consider privacy risks associated with revealing personal data to the infrastructure itself.

Egocentric choice of content: All of the participants adopted an egocentric approach to information selection. Specifically – all participants focussed on selecting information feeds that they thought would be useful to themselves rather than feeds that they considered might be useful to others. We had expected to see a greater focus on using the screen real estate for self expression or promotion.

Openness to Advertising: Three of our participants explicitly flagged portions of their screen as being open to commercial advertising (e.g. the "Sponsored" box on Participant 2's sheet, cf. Figure 6.13a). Later discussions established that these boxes were not there for the financial benefit of the viewer themselves but were typically expected as a feature of modern media.

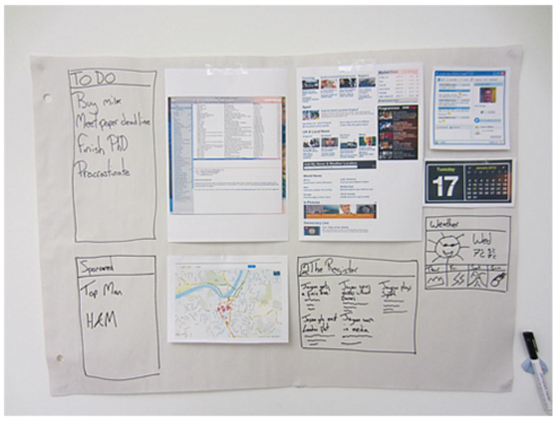
6.3.2 Deployment Experiences

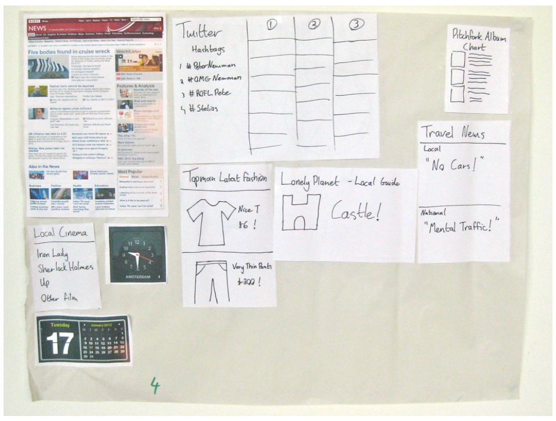
6.3.2.1 Use of Content Descriptor Sets

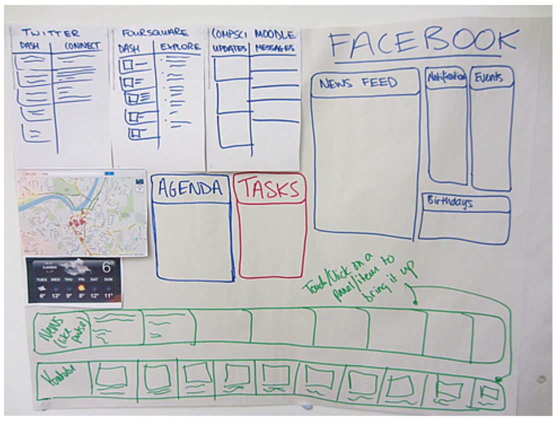
In the following paragraphs we summarise our experiences of using Content Descriptor Sets (CDSs) with Yarely, the e-Channels exporter (described in Section 6.3.2.2), and Mercury. Use of CDSs with Yarely has focussed around reading the files, interpreting the media descriptions and constraints, and identifying changes in the CDS files as the schedules are updated. By contrast, the Mercury and the e-Channels exporter's role as Content Descriptor Factories (i.e. an entity that produces CDSs) has meant that in those systems, the focus has been on the generation of a CDS file.

Describing Media

Content Descriptor Sets have proved to be an effective method of describing the media distributed by the e-Channels exporter and Mercury. The inclusion of a `content-type` string allows Yarely to easily identify the appropriate renderer to be used to play each item, and the `preferred-duration` allows files to be scheduled for an appropriate period of time (e.g. for the duration of a video or for sufficient time to ensure all text has been read). Similarly, the inclusion of one or more `hash` elements allows Yarely to quickly identify whether a file is present within its content cache. Enforcing or encouraging Content Descriptor Factories


(a) Participant 2


(b) Participant 4


(c) Participant 5

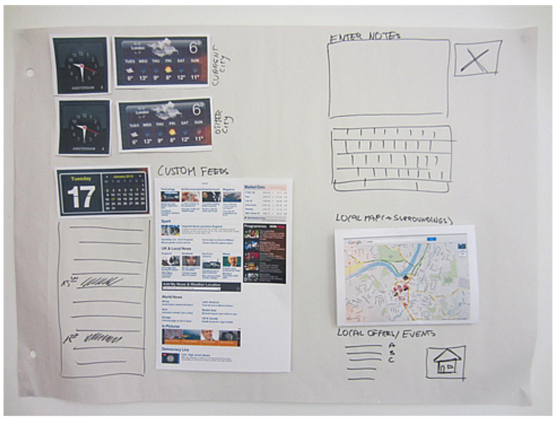

(d) Participant 7

Figure 6.13: Tacita: sample display appropriation screens developed as part of the focus groups.

to populate these attributes and elements reduces the need for equivalent processing at the display node. For example, if the `content-type` string was not required, each individual display node would have to analyse all fetched files to determine if the media type was supported and to select an appropriate renderer – by running this operation once at the Content Descriptor Factory, the information provided in the CDS allows a display node to choose not to fetch media files that cannot be rendered (reducing network traffic) and facilitates renderer selection (reducing processing).

To date all media files exported from both the e-Channels CDS exporter and Mercury have been encoded using the CDS format with no requirements for modification or extension of the schema.

Representing Constraints

The e-Channels exporter provides the widest use of constraints requiring date, time, day of week, and ratio constraints. Mercury requires support for fewer constraints but requires one constraint type not used by e-Channels, that of ordering. All five constraint types are described in the CDS schema definition.

Both date and time constraints use standard XSD data types that provide support for timezones if needed. If no timezone is specified, the date or time defaults to the display's local timezone. This flexibility allows a CDS to encode both an absolute time (e.g. for an international press release) or a local time (e.g. to ensure the lunch menu of a global restaurant chain is shown between the hours of 12 noon and 1pm regardless of the location of the display).

Perhaps the most significant limitation of the CDSs representation of constraints is the lack of provision for identifying when items should be scheduled based on the union of a group of constraints. In the current representation any number of constraints may be applied to a `content-item` or `content-set` element and these elements are assumed to be combined with a logical `and` operation at the display node. To date our experiences with e-Channels, Yarely, and Mercury have not required the complexity of constraint grouping and so we consider that the benefits gained from maintaining a simple approach outweigh the restrictions.

Use of XML as the underlying Document Format
XML is a common form for the interchange of documents over the Internet and API provision for reading and writing XML documents is available in a wide range of programming languages. The use of XML has therefore made it easy to read and write the underlying elements from which the CDSs are formed. Yarely uses the `xml.etree.ElementTree` module in Python 3 to read the document structure and generate a tree of Python objects. The equivalent Python 2.7 module is used in Mercury to convert the Django model objects to CDS elements. In the e-Channels exporter, the PHP DOM extension is used to write the CDS XML.

In addition to its wide-ranging programming support, XML documents support the use of schema definitions to determine if the represented content is valid. Such a schema for Content Descriptor Sets is provided in Appendix A. This schema definition has been a useful tool for development, helping to confirm that Mercury and the e-Channels exporter were generating valid CDS data prior to integration with Yarely.

Finally, the selection of XML as the underlying document format provides benefits in terms of human-readability but can come at a cost due to larger file sizes. Using our most prolific content producer as an example, we measure the file size of the generated CDS for their content. A snapshot of the channel taken in December 2013 contained eleven items (all images) resulting in a CDS 5.9KB in size. By comparison the image files themselves ranged in size from 74.94 to 116.25 kilobytes. We therefore conclude that the verbosity of XML compared some other formats poses no significant concern, and that the benefits of wide-support and human-readability are of greater value than the efficiency of encoding data in a more terse format.

6.3.2.2 Deployment Case Study: Integrating with a Legacy System

Existing display deployments are numerous. The Content Descriptor Set (CDS) format described in this thesis, together with Yarely's openness to accept any number of CDSs is intended to allow the architecture to easily integrate these existing and future third-party systems. In this section we demonstrate the effectiveness of the architecture to integrate with such systems through a case study of adapting one existing system e-Channels to export content in the CDS format.

Section 1.4.4 described e-Channels, a content management system used for day-to-day digital signage content on the Lancaster University campus. e-Channels was originally designed as part of the e-Campus system. To support our deployment of Yarely we created a web-based exporter that converted content channels and subscriptions from e-Channels to a set of Content Descriptor Set documents. This allowed Yarely to maintain compatibility with the existing e-Channels system.

As described in Section 1.4.4, the e-Channel system stores channel structures and media items as a set of directories and files; channel availability is stored in a MySQL database. A PHP web interface allows users to create channels and determine their availability, whilst a CIFS file share is used to manipulate content within each channel. In order to interface with e-Channels we implemented a set of scripts to read the e-Channels database and file system structure and generate a Content Descriptor Set (CDS) file.

The e-Channels CDS exporter is implemented in PHP and served from the same host as the existing Channel System web interface. PHP's `mysqli` package is used for accessing the e-Channels MySQL database. File system operations use PHP's builtin directory and string functions (for reading the file system and calculating file hashes), the `finfo` PHP package (for detecting file types), and the `ffmpeg` command-line tool (for calculating audio and video duration).

Each individual channel is converted to a CDS by the script `per_channel.php` that takes the channel ID as a GET parameter (e.g. `http://e-content/yarely/per_channel.php?channel_id=205`). The combined channel subscriptions for each display is converted to a CDS containing a number of remote `content-set` links by the script `per_display.php` that takes the display ID as a GET parameter (e.g. `http://e-content/yarely/per_display.php?display_id=ecampus-07`.

The e-Channels CDS exporter has been in day-to-day use since March 2012 and successfully outputs the structure and content of e-Channels to valid CDSs. Figure 6.14 shows snippets from the output of the e-Channels CDS exporter. Our Yarely display deployment uses the e-Channels CDS exporter as its primary source of content schedules demonstrating successful integration with a legacy system.

6.3.2.3 Supporting Application Distribution in Open Display Networks

Mercury was designed to act as a distribution platform for display applications, i.e. as one of a number of content sources for future open pervasive display networks. In order to do this, Mercury must be able to integrate existing and future display applications. In this section we evaluate Mercury by reflecting back on the process of adding an initial set of ten applications.

The set of ten initial applications (listed in Table 5.4, page 238) includes three applications created within our own research group, four InstantPlaces applications developed by researchers at Universidade do Minho (see Section 2.5.10, page 88 for an overview of InstantPlaces), two social networking applications created by researchers at the Universitá della Svizzera Italiana (USI), and 'Digifieds', a classifieds application developed by Universität Stuttgart.

Applications From Lancaster

The 'Missing Child' Application The 'Missing Child' application is an example of a location-aware application. In this application, all displays within a increasing geographic distance of a specific location display a simple poster-style alert containing details of a missing child (Figure 6.15a). This application is designed to address the scenario presented in Section 1.2.1 (page 3).

The Missing Child application is a web application, and was first developed to connect with the e-Campus deployment described in Section 1.4. The application consists of two interfaces and a background server process. A display frontend produces the previously described poster alert. A second web interface (Figure 6.15b) provides a backend allowing security staff to create new missing chid alerts, which are stored in a database. Finally, a background server process fetches recent alerts from the 'Missing Child' application database and display locations from the e-Campus displays database. Comparing the locations associated with the alerts against the display locations allowed the background process to produce a set of displays to which the display frontend should be scheduled.

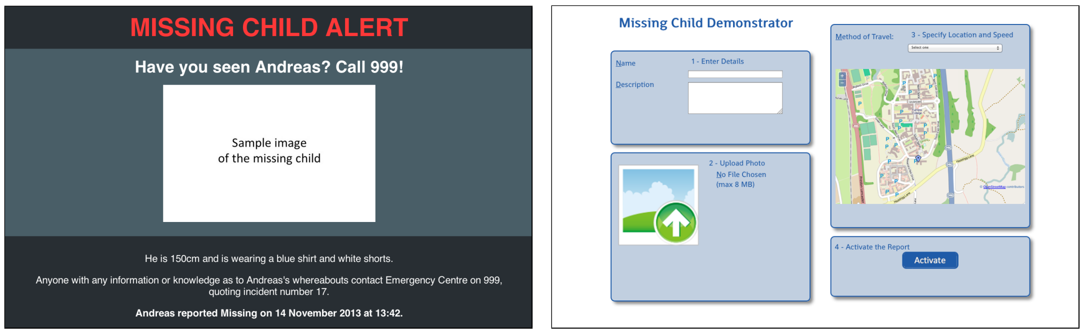
```
<?xml version="1.0"?>
<content-set name="e-Campus Display: ecampus-07" type="inline">
  <auth handler="none" /><feedback />
  <constraints><scheduling-constraints>
      <time><between start="08:30:00" end="17:30:00" /></time>
      <day-of-week><between start="monday" end="sunday" /></day-of-week>
  </scheduling-constraints></constraints>
  <content-set name="SCC - SCC News" type="remote">
    <constraints><scheduling-constraints>
      <playback ratio="0.40" />
    </scheduling-constraints></constraints>
    <requires-file><sources>
      <uri refresh="2 minutes">
        http://e-content/yarely/per_channel.php?channel_id=400
      </uri>
    </sources></requires-file>
  </content-set>
  <content-set name="public space channel news"
     type="remote">...</content-set>
</content-set>
```

(a) Display export

```
<?xml version="1.0"?>
<content-set name="public space channel news" type="inline">
  <auth handler="none"/><feedback/>
  <constraints><scheduling-constraints>
      <playback order="random" avoid-context-switch="false"/>
      <time><between start="00:00:00" end="23:59:00"/></time>
      <day-of-week><between start="monday" end="sunday"/></day-of-week>
      <priority level="medium"/>
  </scheduling-constraints></constraints>
  <content-set name="public space channel news" type="remote">
    <requires-file><sources>
        <uri refresh="2 minutes">
          http://e-content/yarely/per_channel.php?channel_id=205
        </uri>
    </sources></requires-file>
  </content-set>
  <content-item content-type="image/jpeg; charset=binary"
     size="95909 bytes">
    <requires-file>
      <hashes><hash type="md5">921bc01d1607218f0267e63a4c5f0029</hash><hash
          type="sha1">8c997ef8196562d22e995bdea0d1c0e9e8aface2</hash></
          hashes>
      <sources><uri>http://e-content/ChannelContent/Aspiring%20Businesses%20
          in%20Cumbria.jpg</uri></sources>
    </requires-file>
  </content-item>
</content-set>
```

(b) Channel export

Figure 6.14: Snippets of sample XML output from the e-Channels CDS exporter.


MISSING CHILD ALERT

Have you seen Andreas? Call 999!

Sample image of the missing child

He is 150cm and is wearing a blue shirt and white shorts.

Anyone with any information or knowledge as to Andreas's whereabouts contact Emergency Centre on 999, quoting incident number 17.

Andreas reported Missing on 14 November 2013 at 13:42.

Missing Child Demonstrator

1 - Enter Details

Name

Description

2 - Upload Photo

No File Chosen

(max 8 MB)

3 - Specify Location and Speed

Method of Travel:

4 - Activate the Report

Activate

(a) Display frontend (b) Backend

Figure 6.15: Applications added to Mercury: The Missing Child Application

The e-Campus low-level scheduling API is used to schedule the web frontend to the display.

Integration of the 'Missing Child' application was done in parallel with adaptations to move from the e-Campus deployment to Yarely. Since Yarely provides no centralised service for groups of displays, there was no database to query for the display's location – the application store's RESTful API was an ideal replacement. In the version deployed to Mercury, the 'Missing Child' application continues to pull the complete set of displays (as it did in e-Campus) and then calculates the distance from the location associated with the alert. In the long-term this is clearly unscalable, and future additions to the RESTful API could provide a mechanism for querying by display location. Whilst the background server process has been significantly altered, the backend web interface and the application's database remain unchanged. The frontend web interface has been modified slightly to allow cycling through posters in the event that multiple alerts are valid in a single location (the previous version showed the most recent alert only). The frontend web interface was then added to Mercury as a new entry of type URL application.

The 'World Clock' Application The 'World Clock' application is an example of a location-aware application with personalisation support. In this application, the display shows a clock with the local time plus two other clocks with times from random or user-specified locations (as illustrated in Figure 5.10c, page 232).

The World Clock application is a web application, first developed as part of a test suite for Tacita. The application consists of a web server with two interfaces – one for mobile requests, and one for requests for the display content. The application keeps a local database containing the locations of all known displays. Each content request made to the server must include the display URI to allow the location to be found in the local database. 'World Clock' also stores user requests in its local database (keyed by display URI).

In order to integrate with Mercury, we modified the 'World Clock' application such that its mechanism for looking up a display location used Mercury's display ID and RESTful API rather than a display URI and the local database. We also modified the user request submission process so that requests were keyed by Mercury's display ID. Overall a total of five files were changed, with a total of sixty-three line insertions and fifteen deletions. Once all changes had been made, the 'World Clock' application's display interface was then added to Mercury as a new entry of type URL application.

The 'News from Lancaster University' Application The 'News from Lancaster University' application is an example of a traditional digital signage application in which a set of static content items are shown on the display. In this case, the set of items is comprised of media files selected by the Lancaster University Press Office, typically images such as the one shown in Figure 6.17a.

Unlike the other applications added to Mercury in this initial phase, News from Lancaster University is not a web application, but instead is an example of the 'SlideShow' application type (see Section 5.4.2, page 217). This application is a representation of one channel from the e-Channels system described in Section 1.4.4; within the channel system, media items within each channel are stored in a directory on a networked file share.

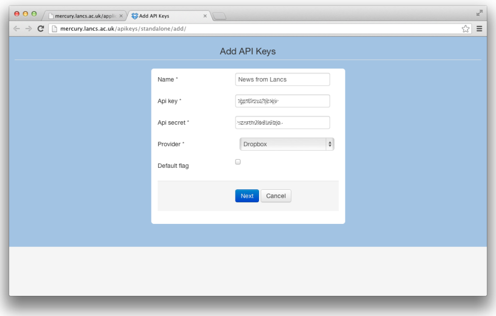
In order to integrate this application into Mercury, the existing networked file share needed to be connected to a Dropbox[1] folder. A Dropbox folder was created and mounted on the e-Channels web server. A crontab entry was then created to ensure regular syncing of the data from the networked file share to the

[1]http://www.dropbox.com

Dropbox folder. The Dropbox folder was then added as a new application using Mercury's add applications wizard.

Due to its explicit provision for SlideShow applications, integration of the existing News from Lancaster University e-Channels content was very straightforward, with no changes made to the underlying e-Channels application. However, mapping files from the existing network file share to a Dropbox folder requires the server to maintain two physical copies of each file and a periodic update to keep the Dropbox in sync with changes made in e-Channels.

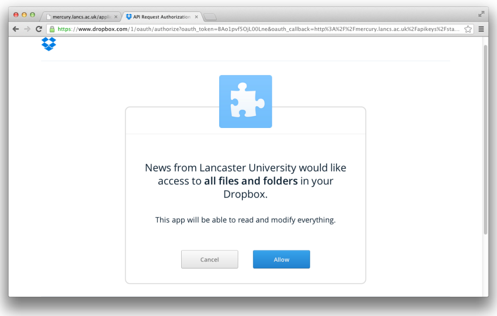
Finally we note that addition of a SlideShow application is slightly more complicated than the addition of a URL application. In order to add the application to the store, the developer must generate a Dropbox API key using the Dropbox website and then paste this key into the store as part of the add application wizard (Figure 6.16a). The store then automatically opens a Dropbox window in which the user must confirm that the store has permission to use the API key (Figure 6.16b).


mercury.lancs.ac.uk/apikeys/standalone/add/

Add API Keys

Name *	News from Lancs
Api key *	[redacted]
Api secret *	[redacted]
Provider *	Dropbox
Default flag	☐

(a) Entering the API key into the add applications wizard.


https://www.dropbox.com/1/oauth/authorize?oauth_token=BAo1pvf5OjL00Lne&oauth_callback=http%3A%2F%2Fmercury.lancs.ac.uk%2Fapikeys%2Fsta...

News from Lancaster University would like access to **all files and folders** in your Dropbox.

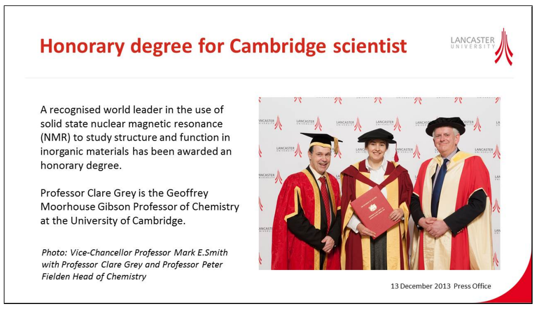
This app will be able to read and modify everything.

(b) Confirming access to the Dropbox folder through the API key

Figure 6.16: Mercury: Addition of a Dropbox Slideshow Application.

Instant Places Applications

The InstantPlaces applications consist of four location-aware applications that share a common backend. The applications also support differing degrees of viewer personalisation. The InstantPlaces backend is focussed around the concept of place, each place may contain one or more displays.



Honorary degree for Cambridge scientist

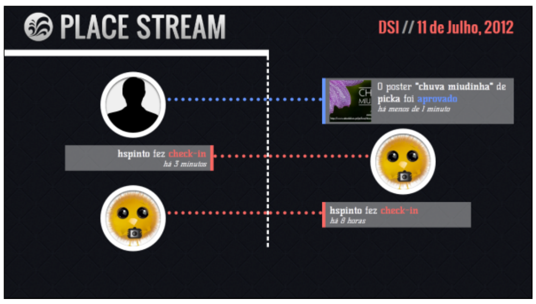
A recognised world leader in the use of solid state nuclear magnetic resonance (NMR) to study structure and function in inorganic materials has been awarded an honorary degree.

Professor Clare Grey is the Geoffrey Moorhouse Gibson Professor of Chemistry at the University of Cambridge.

Photo: Vice-Chancellor Professor Mark E.Smith with Professor Clare Grey and Professor Peter Fielden Head of Chemistry

13 December 2013 Press Office

(a) News from Lancaster University



PLACE STREAM

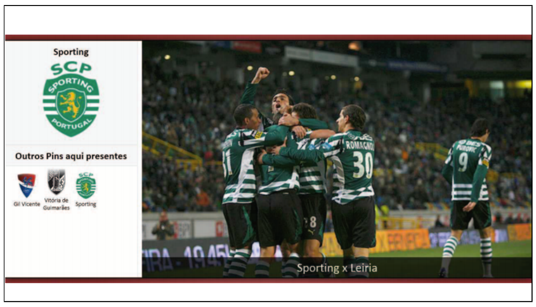
DSI // 11 de Julho, 2012

O poster "chuva miudinha" de picka foi aprovado
há menos de 1 minuto

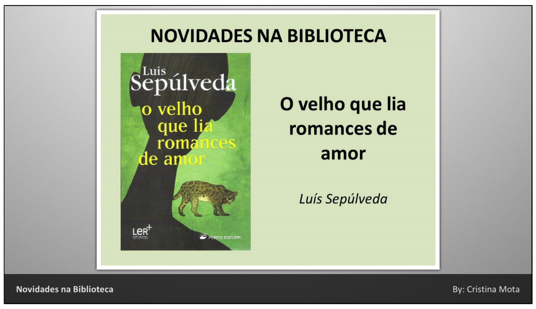
hspinto fez check-in
há 3 minutos

hspinto fez check-in
há 8 horas

(b) Instant Places: Activity Stream



(c) Instant Places: Football Pins



NOVIDADES NA BIBLIOTECA

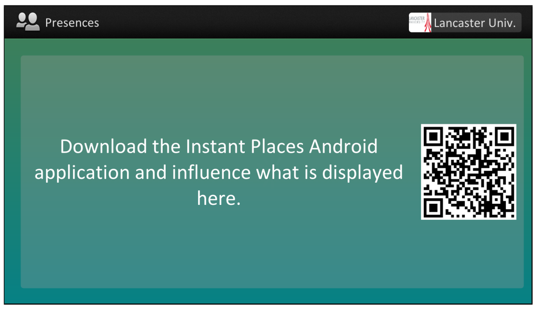
O velho que lia romances de amor

Luís Sepúlveda

Novidades na Biblioteca

By: Cristina Mota

(d) Instant Places: Posters

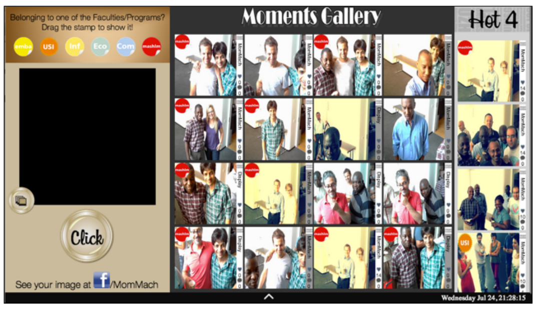


Presences

Lancaster Univ.

Download the Instant Places Android application and influence what is displayed here.

(e) Instant Places: Presences


Belonging to one of the Faculties/Programs? Drag the stamp to show it!
Click
See your image at /MomMach
Moments Gallery
Hot 4
Wednesday Jul 24, 21:28:15

(f) Social Networking: Moment Machine (left) and Moment Gallery (right)

Figure 6.17: Applications added to Mercury

The ‘Posters’ application (Figure 6.17d) shows viewer-submitted posters (provided that those posters have been approved by the display owner). The ‘Presences’ application (Figure 6.17e) shows the set of mobile application users who have recently checked-in to the place in which the display is located. The ‘Football Pins’ (Figure 6.17c) application shows images and information about football teams of interest to display viewers. Finally, the Activity Stream application (Figure 6.17b) provides an overview of recent interactions with other InstantPlaces applications in that place.

All four applications are implemented as web applications. The applications are tightly-coupled to the backend and require a place ID to be specified in order to generate content. Some applications also support additional parameters such as transition intervals and effects.

In integrating the InstantPlaces applications to Mercury, no modifications to the applications were made. However, deploying any InstantPlaces application to a display currently requires the creation of a new place in the InstantPlaces backend. To do this, a display owner must register with the InstantPlaces system (separately to their Mercury registration) and create a new place entry, duplicating information that they will have already contributed to the application store. This additional step increases the burden for display owners.

Each InstantPlaces application is added to Mercury as a new entry of type URL application.

Social Networking Applications from USI

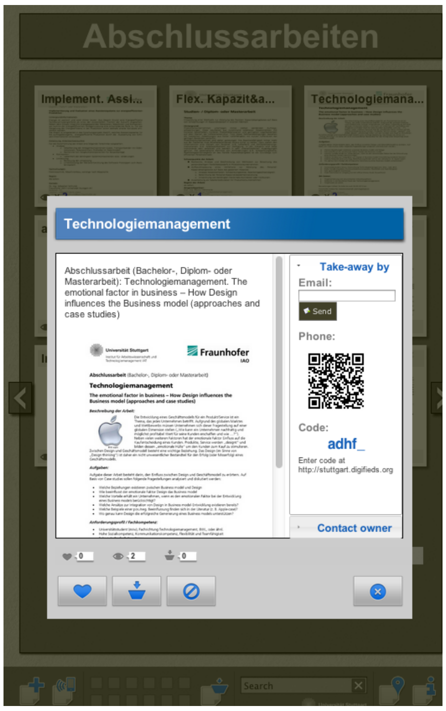
‘Moment Machine’ is an interactive (touch-screen) application that uses a camera associated with a display to allow viewers to take photographs and post them to Moments Gallery and to popular social networks. ‘Moments Gallery’ shows photographs that have been posted using Moment Machine. Unlike many of the applications described in previous sections, Moment Machine and Moments Gallery are not location-aware – all images are posted to all displays.

Both Moment Machine and Moments Gallery are web applications consisting of a display interface (shown in Figure 6.17f) and a shared backend server to store the images.

In integrating the two applications with Mercury, we note that the applications themselves make considerably greater demands of the displays than the previous examples. Unlike other applications, Moment Machine is designed to be shown on a portrait display. In fact, the two applications are intended to be shown on the same display simultaneously (as in Figure 6.17f) requiring space-multiplexing at the display. The Moment Machine application also requires a camera and touch-screen capabilities and Moments Gallery also benefits from touch interaction.

Moment Machine and Moments Gallery were originally developed for display on a single display at Universitá della Svizzera Italiana – for this reason, neither application is designed with location-awareness, but instead images are held in a single pool shared by all displays. On deployment to Mercury, it became clear that the application currently has little value for display owners in a larger network since the images will be from a number of unrelated locations.

Neither Moment Machine nor Moments Gallery have been modified to allow integration with Mercury. Both applications have been added to Mercury as new entries of type URL application. We note however that the lack of any modifications to this application have limited its value in a large display network.



Abschlussarbeit (Bachelor-, Diplom- oder Masterarbeit): Technologiemanagement. The emotional factor in business – How Design influences the Business model (approaches and case studies)

Figure 6.18: Applications added to Mercury: The Digifieds Application

Digifieds

'Digifieds' is an interactive location-aware application (also described in Section 2.2.7.2, page 51) that shows virtual boards of classified adverts created by display viewers and users of the Digifieds mobile application (Figure 6.18).

The Digifieds application is a web application, first developed for deployment on the UBI-hotspots in Oulu, Finland.

The application consists of a backend server and the web-based display interface. The backend maintains a set of instances, collections of boards relating to a specific community or location. Each classified advert is posted to a specific instance. Within the display interface, the instance ID is a required parameter used to filter the adverts to be shown.

Digifieds requires a high-definition touch-screen display and portrait orientation. Like the InstantPlaces applications, Digifieds is tightly-coupled to its back end and requires the creation of a new instance before the application can be deployed to a new location. However, unlike InstantPlaces which allows place creation through a web interface, creation of a new instance can only be done by a Digifieds system administrator creating a significant barrier to deploying the application in new locations.

No modifications to Digifieds were made for integration with Mercury. Digifieds has been added to Mercury as a new entry of type URL application.

Summary

Most applications required some modification before integration with Mercury. Typically, the applications maintained a backend or local database that provided information about the displays to which the application had previously been deployed. For example, the World Clock application kept a local database to map from known display URIs to geographic locations, and the InstantPlaces applications each connect to the backend to identify the location and current set of mobile users associated with the place in which the display is located. This common design pattern is inevitable for multi-display, location-aware, applications designed outside the context of an application store.

Two of our contributed applications had originally been designed for a single display. These applications posed a contrasting problem as they had no location-awareness and made little sense outside of their original environments. Whilst the overall pattern of the displays was generalisable, no mechanism had been provided to allow display owners to customise the application by specifying a display location, social media account or other content filter.

Finally, we note that some of our integrated applications required specific display hardware and most applications made assumptions about the screen orientation and/or resolution. At present these requirements are not represented within Mercury and a display owner must try the application for themselves in order to determine whether or not it functions correctly at the display.

Based on our experiences with Mercury, we anticipate that future multi-display applications designed for distribution through a shared channel such as Mercury could take advantage of APIs provided by these channels to gather data about a display and avoid the need for the storage of display metadata in additional backends. Similarly, we anticipate that over time our existing application set would be gradually modified to replace backend display metadata storage with calls to the Mercury API. Our current API supports the most common use case for existing backends, that of fetching a display's location and work is in progress to add data about orientation and input devices. Applications may still need to maintain backends for storage of application-specific data (e.g. viewer preferences, poster content) but common data about the displays themselves and the environment in which they are sited can be offloaded to Mercury.

6.4 Limitations

The quantitative evaluation independently measured performance aspects of the three components of our architecture. The resulting data provided a number of insights that have been described earlier in this chapter. However, no evaluation is exhaustive and sections below highlight key limitations of our quantitative and qualitative work.

6.4.1 Limitations of the Quantitative Evaluation

Focus on a subset of Yarely operations. The studies of Yarely's performance focussed on two key aspects – rendering (demonstrating that a range of media could be displayed on the screens) and resilience (demonstrating the effect of caching on reducing network demands). Instrumenting Yarely as a whole could provide additional insights (e.g. providing a profile of how responsive Yarely is to subscription changes, how items are swapped in and out of cache, and how accurately scheduling constraints are adhered to).

Absence of quantitative evaluation of Content Descriptor Sets. Our CDS evaluation focused on measuring the effectiveness of the document format for representing the required data through reflection on deployment experiences. The suitability of the CDS format is difficult to study quantitatively but a simple set of measures could compare the CDS with equivalent files for other popular formats (e.g. SMIL) in terms of file size generated, network transfer time etc.

Semi-realistic Mercury application counts. Our Mercury profiling demonstrated the performance of a subset of the RESTful API calls given a varied number of applications in the database (0–14,999 for calls related to the creation of new objects; 0–1,499 for retrieval operations). Whilst current display deployments typically have very small numbers of applications, the introduction of application stores to this domain is predicted to increase the number and variety of applications available [DLJS12]. Current mobile application stores contain one or two orders of magnitude more applications than the maximum number studied by our measurements [1].

Non-realistic Mercury review and purchase counts. Similarly, our experimental limitations of one billing model, one purchase, and one review per application are not representative of current behaviour in mobile application stores – downloads across the iOS and Android stores are in the billions per month, and at the end of 2013, applications had an average of one review each in the App Store and two reviews each in the Google Play store [Row14].

Measurement of non-concurrent Mercury requests. We do not provide a study of the Mercury performance when in use by multiple concurrent users. We consider the prevalence of Django in underpinning real-world websites as sufficient evidence that Mercury could handle both concurrency and load. However, existing tools such as `ab` [The14] could be used to measure the

[1]BlackBerry World reported 235,000 Blackberry applications available in July 2013 [Bla13], over 1 million applications were reported to be available in the Apple App Store in October 2013 [App14b], and the Google Play store was reported to contain 1,208,201 Android applications on May 11 2014 [App14a].

performance of Mercury when under load.

Small Set of Environments used to study Tacita Proximity. The Tacita evaluation was conducted on a subset of two environments taken from the larger Yarely deployment. The locations were deliberately selected for their difference in layout in terms of visibility and surrounding walking routes; nevertheless we note that a set of two environments cannot represent the full range of display deployments and further evaluation with a wider set of locations could provide better indications of content exposure and accuracy.

6.4.2 Limitations of the Qualitative Evaluation

Where the quantitative studies were focussed predominantly on individual elements of the architecture, the qualitative evaluation work

Limited scale and scope of display owner evaluations. Our interviews with Display Owners included six individuals all recruited from within Lancaster University. Although the interviewees varied in terms of spaced owned (including commercial spaces), we note that the small-scale and limited scope restricts the applicability of the results.

Focus on predicted use. The qualitative evaluations with both display owners and viewers sought to identify participants attitudes to the deployment of future appropriation mechanisms; in some cases these were supported by prototypes. One clear limitation to the current evaluation is a lack of longitudinal deployment of the iterated architecture with data for all three systems (Yarely, Mercury, and Tacita) the content descriptor set data that shows actual behaviour and attitudes with reference to a real-world deployment.

6.5 Revisiting the Requirements

6.5.1 R1. Support for day-to-day signage functionality

Our first requirement [R1, page 142] was concerned with provision for *day-to-day digital signage functionality* and is addressed by the Yarely component (for media scheduling and playback) and by the Content Descriptor Sets (CDS) format for describing media and constraints. The suitability of these components for providing a day-to-day signage platform is evidenced in the long-standing deployment (approximately two years) of the system at the Lancaster University campus.

Both prior research [BIF+04] and our own evaluation of files contributed to e-Channels [Section 3.3.3.2] suggest that images, vido and web content are key media for digital displays. Yarely supports playback of images and web content on all platforms with additional support for video and other media types on Mac OS X. Overall, Yarely can support 87.24–95.09% (depending on platform) of the files analysed in the e-Channels snapshot described in Section 3.3.3.2; e-Campus supported 94.71% of these files. We therefore conclude that Yarely successfully delivers support for playback of traditional media items (R1.1).

Representation of date-, time- and priority-based scheduling constraints (R1.2) in the CDS structure is described in Section 5.3.2.1 and their implementation in the Yarely scheduler in Section 5.2.2.1. Finally, Yarely's provision for power management of physical display hardware (R1.3) is described in Section 5.2.2.1.

Based on the described implementation and deployment we conclude that the architecture successfully provides support for day-to-day signage functionality through the Yarely component and the CDS document format.

6.5.2 R2. Compatibility with other systems

Our second requirement was to provide *compatibility with existing systems and connection points for third-party systems.*

A number of design features are intended to ensure compliance with this requirement. The CDS format provides an extensible cross-platform mechanism for describing content availability and is explicitly intended as a method of connecting components within the architecture. Existing third-party playback components could be compatible with other portions of the architecture by accepting the CDS as input; similarly existing media providers could be integrated by providing an exporter that generates a CDS. Compliance with other systems is also designed into Tacita through the map provider component, which caches display capabilities such that existing display infrastructure without broadcast technology can still advertise their capabilities. Finally, the purpose of Mercury is to provide a distribution channel for display applications including those developed by other parties.

The effectiveness of these components to deliver connection points and compatibility has been demonstrated through the following:

- Section 6.3.2.2 provided a case study of the integration of Yarely with the existing system e-Channels using the CDS format.
- Section 5.6 described a demonstration [XCDS13] of the integration of Yarely and Tacita with Internet Suspend/Resume [KS02].
- Section 5.6 and Table 5.4 described a set of display applications that have been contributed to Mercury including seven existing applications from third-parties. The process of integrating these applications into Mercury was presented in Section 6.3.2.3.

6.5.3 R3. Flexible Initiation of Appropriation Requests

Requirement three called for the architecture to *allow for user detection, request submission, and interaction through a range of existing and emerging hardware.* The requirement was broken down into three additional sub-requirements:

- R3.1 - The architecture should support user detection and request submission though contemporary mobile phone technologies.
- R3.2 - Our architecture for display appropriation should anticipate future changes in hardware and be designed to support a heterogeneous set of sensing and interaction methods.

- R3.3 - Our entire pervasive display architecture should be open to a range of current and emerging hardware types, operating systems and platforms (i.e. not just sensing technologies, but also display hardware and network protocols).

R3 requirement is addressed in the general case by Yarely's extensible architecture, and more-specifically by providing a sample request and user detection mechanism through Tacita.

Within Yarely, the Sensor Management component provides support for sensing data through socket connections and HTTP requests – if a viewer did not want to use Tacita for submitting their appropriation requests, the HTTP request sensor would be a viable alternative for supporting submit viewer appropriation requests from personal mobile devices (R3.1). More likely though is the use of Tacita mobile application which addresses R3.1 by supporting a range of proximity detection mechanisms (evaluated in Section 6.2.3) to identify when a display appropriation request should be made. The requests themselves are transferred over the Internet and can be sent over any available connection (e.g. 3G, WiFi).

Yarely's design allows for easy addition and substitution of components in response to changing hardware and developments in sensor and interaction technologies (R3.3). For example, the HTTP server sensor handler was not part of the original system deployed in March 2012, but was added later in order to support the demonstration given at PerDis 2013. This same component-based design also allows support for platform-specific components, allowing Yarely to execute on multiple platforms as demonstrated by our provision for Mac OS X and Linux (R3.3).

Across the architecture as a whole, our use of multi-platform programming languages, protocols, and libraries help to ensure that the majority of the software is capable of running on a range of hardware types and platforms (R3.3). The Tacita mobile application remains the only component with limited platform support – in this case two applications have been developed to provide the same functionality, providing coverage on the two most popular mobile operating systems.

6.5.4 R4. Support a range of appropriation usage models

Requirement four was concerned with support for a range of appropriation usage models, with a specific focus on supporting the three models identified in Chapter 3: *walk-by*, *active* and *longitudinal*, appropriation (R4.1). Providing support for such models required the architecture to *provide mechanisms by which flexible scheduling (duration, interruptability constraints) can be supported* (R4.2) and to consider how personalisation requests from multiple simultaneous viewers could be accommodated (R4.3).

The proximity detection methods used by Tacita are shown to be effective for walk-by appropriation in Section 6.2.3, achieving non-zero content exposure for nearly all detection methods individually, and providing content exposure for up to 98% of the time that the viewer could see the display with multiple detection methods in combination. Our demonstration of Tacita and Yarely used in combination with Internet Suspend/Resume at PerDis 2013 [XCDS13] focussed on the use of the architecture for active appropriation and showed the suitability of the architecture for supporting scenarios such as that of Dr. Jones, previously presented in Section 3.2.2. We believe that Tacita is also appropriate for longitudinal appropriation. By increasing the size of the trigger zone Tacita preferences can be distributed over a wider geographic area. Pairing this with a more relaxed approach for making requests from the display applications for screen time allows Tacita to support longitudinal appropriation.

In order to support appropriation requests, our Content Descriptor Set format provides a range of constraint types and our Yarely Scheduling and Management component resolves these constraints to select an item to play. One key constraint used in our Tacita and Missing Child systems, is the provision for priority levels (addressing requirement R4.2). In addition, the constraints can also be used to determine the duration or physical boundaries of content on the screen allowing representation of time- and space-multiplexing in order to help overcome the challenge of scheduling multiple personalised applications simultaneously (R4.3). Yarely and Tacita both also give consideration to this problem – Yarely's Scheduling and Management component provides support for space-multiplexing of viewer-driven content for up to twenty concurrent users, whilst many of our Tacita display applications use aggregation to create a single content item the matches the preferences of multiple co-located display viewers.

6.5.5 R5. Minimise security and privacy concerns

Requirement five called for the architecture to *minimise security and privacy concerns*.

Chapter 3 identified two broad areas of privacy concern: disclosure through the construction of tracks and profiles and disclosure through the display or inappropriate or revealing content. Our evaluation in Section 6.3.1.2 showed that many users have some awareness of the latter concern, choosing not to use display appropriation for social networking content – particularly when reminded that they may view the screens in the presence of others. Our personalisable applications also reduce the risk associated with this privacy concern by deliberately aggregating requests from multiple users with some random content – this provides some plausible deniability since content cannot be matched to any individual even when very few viewers are collocated with the display.

In order to address the second concern, the design of Tacita allows viewers to make appropriation requests whilst deliberately limiting the disclosure of personally identifiable information (i.e. profiles) to the display infrastructure. Our focus groups highlighted viewers' simplistic models of information sensitivity in which privacy concerns were focussed entirely on the personal data that other viewers may see on the displays rather than the risks associated with revealing personal data to the infrastructure itself.

Finally, the support for VNC connections through a web-based Tacita application (*ubiVM*), and through Yarely's VNC renderers provides a mechanism for users to execute custom applications without compromising the security of the displays themselves. This ensures that viewer applications do not impact the reliability of the display.

6.5.6 R6. Support a wide range of applications

Requirement six specified that the architecture should provide *support for a wide range of applications executing in different locations and from multiple content* *sources*. This is important to provide flexibility in the type of applications that can be used for appropriation. The requirement is addressed by Yarely and Mercury.

Current display applications typically take the form of a web page with a server to manage application data and logic. Support for such applications is supported by Yarely on each of our three target platforms. Such applications can also be added to Mercury through the URLApplication type. As described in Section 6.3.2.3, applications of this kind deployed to Yarely through Mercury include a mixture of applications developed at Lancaster and those developed by other research groups (e.g. Moments Gallery).

Support for emerging applications can be provided through the extension of the existing architectural elements (e.g. by adding new renderer components to Yarely or providing an additional Application subclass for Mercury. Alternatively, support for custom applications types can already be achieved through use of either Tacita's *ubiVM* application or Yarely's provision for VNC rendering. This provision for VNC rendering also provides a mechanism for allowing application execution in a range of locations. For example, the demonstration given at PerDis 2013 [XCDS13] and described in Section 5.6 used Yarely's VNC renderer to connect back to one of two virtual machines (VMs). One VM was instantiated in the cloud, many hundreds of miles from the display, a second VM was instantiated on the display node itself. Executing applications on each of the VMs produced different performance characteristics (as in our earlier cloudlet study presented in Section 3.3.2.

6.5.7 R7. Separation of content creation and display ownership

Our final requirement was based on experiences of the e-Channels system and called for the architecture to *maintain a distinction between content creators and display owners*. This requirement is addressed through Content Descriptor Sets (CDSs) and the display application store.

Our overall architecture with CDSs created by Content Descriptor Factories provides a natural method of separating the process of owning a display from that of creating and distributing content. Content providers can export their content using one or more CDSs hosted by a Content Descriptor Factory. In order to select content items to show at their display, owners simply need to maintain a list of interesting Content Descriptor Factories to which they subscribe their displays.

Mercury is one such example of a Content Descriptor Factory with the explicit goal of distributing content from a range of developers to many displays. Like e-Channels, the application store offers display owners the opportunity to contribute content for their own (and others') displays, but usage patterns from similar application stores in the mobile device domain suggest that in the most common case display owners will simply select content from the existing applications submitted by others. This distinction between content creation and display ownership is critical to systems that support appropriation because in the general case applications and content will be selected (and possibly created) by the viewer rather than the display owner.

6.6 Summary

Together Yarely, Content Descriptor Sets, Tacita and Mercury form an overall architecture for supporting viewer appropriation in future open display networks. In this chapter we have presented an evaluation of both the integrated architecture and its individual components.

Figure 6.1 provides a clear indication that the elements connect together well, with multiple interfaces connecting the systems. Individually, the components demonstrate good performance:

- Our quantitative evaluation of Yarely demonstrated the ability of the system to render a wide-range of media items. The use of ØMQ for interprocess communication was shown to be efficient, taking approximately 0.003 seconds for child processes to register with their parent. Overall processing time associated with rendering items was shown to vary significantly with the renderer type and, in some cases, with file size. The quantitative evaluation of Yarely also demonstrates the effectiveness of the media caching to reduce network traffic and improve reliability; this evaluation is supported by experiences from our demonstrations and deployment in which we have observed continued operation during network failures.

- Quantitative evaluation of Mercury shows that most API calls complete in a timely manner, and in most cases, little performance impact is seen as the number of applications within the store grows.
- Finally, measurements of Tacita demonstrate that using location technologies accessed through a viewers personal mobile device can support even the most time-critical appropriation usage model, that of walk-by appropriation.

Using user studies to validate the combined architecture, we find evidence to show that, in the right circumstances, viewer appropriation can be acceptable to display owners, and that display owners are typically quick to understand both the benefits of display appropriation for passing viewers and the potential for improving their space through provision for display appropriation. Our online survey and focus groups elicited a number of use cases, primarily focussed on utility applications, with a good overlap with our developed application set (e.g. news, weather, clock). These studies also provided evidence that whilst many viewers are aware of the risk of disclosure through the display or inappropriate or revealing content and can take appropriate action to reduce the risk (e.g. by not requesting social media content), understanding of other privacy concerns is limited. We believe this validates our design decision to ensure openness to multiple viewer appropriation mechanisms through Yarely Sensor and Schedule Management components. This openness allows us to use the Tacita mobile clients to avoid disclosing personal identifiers and data to the display infrastructure (instead routing personalisation requests through trusted application providers).

Reflecting back on our deployment experiences of adapting e-Channels data to a set of CDSs and similar experiences from working with CDSs in Yarely and Mercury we conclude that CDSs are an effective method of representing both media items and constraints and that the inclusion of meta data about content items (e.g. file hash, media type) is particularly useful for optimisations at the display node. The underlying XML format is human-readable but the resulting files are not large enough to cause any real concerns in terms of network transfer time.

Our deployment experiences also provide evaluation regarding the suitability of the architecture for integrating with third party systems and display applications. Our case study of integrating Yarely with e-Channels a legacy system previously used as part of e-Campus demonstrates the suitability of Yarely for playing content from any third-party content provider capable of describing its media and constraints in a Content Descriptor Set (CDS). Within the context of Mercury, we demonstrates the successful integration of ten applications into the store and highlight the role of the RESTful APIs as a gradual replacement for existing back-end systems that maintain multiple copies of display metadata.

The overall architecture fits well with our original requirement set, successfully providing both day-to-day digital signage functionality (R1), a wide range of applications (R6) and a range of appropriation usage models (R4), whilst maintaining compatibility with a wide range of hardware types (R3) and legacy and third-party systems (R2). The architecture provides mechanisms to minimise security and privacy concerns (R5), and uses the division between Content Descriptor Factories and Display Nodes to allow separation of content creation and display ownership (R7).

Chapter 7

Conclusions

7.1 Overview

In this thesis we have presented the motivation, design, implementation and evaluation of an architecture to support user appropriation of pervasive display systems. We began in Chapter 1 by acknowledging the prevalence of digital displays in public spaces and by noting that, despite the development of innovative sensing technologies and interaction mechanisms, the majority of real-world deployments follow a broadcast model, in which viewers of the display have no influence on the content shown.

In Chapter 2 we provided detailed background and motivation for our work, in which we surveyed research and systems (with a particular focus on those that allowed users to influence the content appearing on a display). We summarised recent research regarding audience behaviour around public displays and suggested that allowing viewers to appropriate displays could provide an opportunity to increase attention in the open display networks envisaged by Davies et al. [DLJS12].

We continued in Chapter 3 by mapping out the design space for user appropriation/personalisation of pervasive public displays. We characterised the space through a set of scenarios and probes covering a wide range of usage patterns for appropriation of displays in public spaces. From this design space mapping we provided a set of requirements for systems supporting user appropriation of pervasive displays. In the following chapter (Chapter 4), we outlined an architecture to comply with our elicited requirements. The architecture allows displays to be open to content from multiple sources and supports display owners and viewers in selecting content to be shown on a display. The architecture features three key software systems: Yarely, a client-side media player with provision for viewer-driven content; Mercury, an application store to support content distribution in large open display networks; and Tacita, a system for creation and propagation of display appropriation requests by a mobile user. Content schedules exchanged between the systems take the form of a Content Descriptor Set.

In Chapter 5 we described the realisation of our Content Descriptor Sets as XML document format, and the implementation of the three core architectural components identified in Chapter 4. Implementation of Yarely used Python to produce a multi-platform system for content scheduling media playback and the ØMQ messaging libraries to support easy addition of new components in a variety of programming languages. The application store, Mercury, was developed using a range web technologies including HTML, CSS, REST, JQuery and Django, and provided a user interface and RESTful API to provide a range of functionalities to support display owners and content providers in open display networks. Finally, the implementation of Tacita provided two mobile clients (developed using the Android and iOS SDKs) and a set of personalised applications (developed using a range of web technologies). The functionality of all three systems has been proved through a set of deployments and demonstrations, including a long-term deployment of Yarely that spans over thirty displays across four European universities.

We evaluated our systems and their integration to form an overall architecture in Chapter 6, using a mixture of reflections, user studies, surveys and technical measurements. Our evaluation demonstrated the effectiveness of Yarely as a system for rendering content even in periods of disconnectedness, and confirmed its suitability for integrating with third-party systems through the integration of the legacy e-Channels system. Our experiences in integrating with this system, together with further experience using CDSs in Mercury indicate that the Content Descriptor Set format is well-suited to encapsulating content and constraints for exchange between portions of the architecture. Our user studies highlighted the difficulties many display viewers have in identifying the privacy risks associated with revealing behaviour patterns and interests to display infrastructure, validating the design decision for Tacita. Surveys and interviews with display owners also highlighted their openness to appropriation if this has the benefit of increasing the attractiveness of the space in which the display is located. Our deployment experiences with Mercury have highlighted the role of the RESTful APIs for allowing developers to access display metadata, whilst performance metrics show that the API calls complete in a timely manner, and in most cases, demonstrate minimal performance degradation as the number of applications within the store grows.

In this final chapter, we conclude the thesis by highlighting the contributions of the work and identifying those areas that may be fruitful for future research.

7.2 Contributions of this Thesis

This section reviews the main contributions of the work described in this thesis. The sequence in which the contributions are listed is a reflection of the order in which they appear in this thesis and does not imply any ranking by importance.

The contributions are focussed around two key areas. Our first set of contributions relate to the identification of appropriation support as a key issue for future pervasive display networks. Specifically, in this thesis we have contributed:

C1 *A detailed survey of research into pervasive display systems.* This review mapped research from the early 1980s through to the present day (2013); classifying systems according to a number of themes. The review demonstrates the breadth of work in the area and highlights the increasing significance of digital displays in public spaces.

Our specific contribution is a focussed exploration of personalisation and appropriation support within previous pervasive display research. The review indicates that whilst some provision for viewer personalisation has been included in a small set of works, it is not typically made available in today's digital display deployments.

Finally, the review brought together previous studies of audience behaviour around pervasive digital displays to provide a consolidated view.

C2 *Understanding of user attitudes to appropriation.*

- **C2.1** An analysis of the impact of explicitly providing mechanisms for sharing content within a display network, demonstrating that, in certain circumstances, display owners are willing to accept unseen content for their display.
- **C2.2** Exploration of factors that influence the trust that viewers place in content shown on a display infrastructure.

C3 *Detailed usage models and requirements for an appropriation architecture.* We provide a set of requirements for system support for appropriation in open display networks based on reflections around scenarios and experimental probes. Our wide-ranging set of scenarios illustrate a future in which display appropriation is commonplace and serves a range of functions, whilst our experimental probes explore a range of aspects including performance and usability to extend current understanding of the design space.

- **C3.1** We identify three usage models for display appropriation: *walk-by personalisation*, *longitudinal personalisation* and *active personalisation.*
- **C3.2** We demonstrate the practical differences between different models for display appropriation through two implemented systems: use of Bluetooth device names to support walk-by personalisation, and use of virtualisation to support active personalisation.

The second set of contributions relate to the design, implementation and evaluation of an open pervasive display architecture that explicitly supports personalisation/appropriation. Specifically, in this thesis we have contributed:

C4 *A unified pervasive display architecture with explicit support for display appropriation.* We present the design, implementation and evaluation of an architecture for supporting user appropriation and personalisation of future public display networks, identifying a division between components that are co-located with the display and serve a single installation (i.e. those that reside in the Display Segment) and components located within the network for the purpose of serving multiple displays (those in the Network Segment).

The design and implementation of this architecture was comprised of four key components:

C4.1 *A scheduler and media player for open display networks.* We describe the design, implementation, deployment and evaluation of Yarely, a multi-platform component-based scheduling and playback software system developed in Python and designed to run in the Display Segment.

The software's component-based design uses distinct modules to provide the core functions of a signage player (e.g. Subscription Management, Sensor Management, Playlist Generation and Scheduling), and ensure extensibility for changing hardware and software requirements.

A deployment of around thirty displays with over eighteen months duration demonstrated Yarely's suitability as a day-to-day digital signage platform, and a number of demonstrations at conferences and other venues validate its appropriateness for supporting viewer appropriation and complex media.

C4.2 *The design of an application store for distributing content within open pervasive display networks.* We detail the design, implementation and evaluation of Mercury, a Django web software to support the submission of content and its distribution to pervasive displays. Our specific contribution is the design of this application store including the design considerations, underlying data storage model, and user interface workflows.

The Mercury service built upon lessons learnt from the application stores for personal mobile devices in which the openness created by introducing the stores lead to innovation in application development. In Mercury we have a service that encourages a separation of roles between those who create content and applications (e.g. advertisers, designers and developers) and those who consume it (i.e. display owners).

C4.3 *A mechanism for describing content availability within open pervasive display networks.* We define a document format to allow the representation of sets of content items and the constraints that restrict their playback at a display node. The format uses a recursive structure to allow content from multiple sources to be merged into a single tree and supports a wide-range of content types (e.g. images, video, web) and constraints (e.g. time, date, priority, ordering).

These Content Descriptor Set (CDS) documents may be produced by a range of services including existing content repositories and orchestration services, the application store described in C4.2, and future content production or distribution systems – together we term these services Content Descriptor Factories.

Our CDS documents are agnostic of any network protocols used to exchange them.

C4.4 *Support for viewer display appropriation requests.* We provide an integrated architecture that allows display viewers to identify, and send personalisation requests to, displays in their environment. Support for viewer appropriation is provided through support for changing content schedules described using the Content Descriptor Set format, and through a component-based player design that allows integration of modules to support a range of sensor types.

We identify a set of interesting applications for display appropriation based on studies with display viewers and provide implementation of a subset of these identified applications as part of Tacita. Our applications use a variety of techniques to allow support for multiple viewer appropriation request simultaneously including request aggregation and screen multiplexing.

Finally, integration with the Tacita mobile clients (designed by Thomas Kubitza and Christopher Winstanley) provides a mechanism for viewers to appropriate displays within our architecture whilst withholding identifying information from the displays.

C5 *An evaluation of the integrated software architecture* that demonstrates that together our developed systems meet the requirements, presented in Chapter 3, for a software infrastructure for appropriation support in open pervasive display networks.

7.3 Future Work

7.3.1 Deployment and Evaluation of the Integrated Architecture

Section 6.4.1 identified a number of limitations of the evaluation presented in Chapter 6. Most significantly, whilst Yarely has been the focus of a longitudinal deployment, evaluation of Tacita and Mercury has centred on performance measures and short-term trials.

Developing the existing Yarely deployment to source content from Mercury would allow a realistic dataset to be gathered – use of this dataset in future evaluations would overcome the identified limitation that current performance measures were conducted with *non-realistic Mercury review and purchase counts*. Furthermore, encouraging more non-expert users to use the store would allow assessment of the usability of Mercury and its effectiveness in serving multiple concurrent requests.

Integration of Tacita into pervasive display deployments on a long-term basis also offers potential for extending the existing evaluation. Introducing regular viewer interruptions into the Yarely schedule allows exploration of how the architecture works in situations where the day-to-day signage constraints (time, ordering, ratio) must compete with dynamic constraints such as viewer presence. The deployment would also highlight the effectiveness of the circle sectors for triggering content in response to viewers (both content exposure and accuracy) and the effort involved in defining such sectors for a larger and more varied deployment than that used in the evaluation presented in Section 6.2.3.

7.3.2 Content Moderation in Open Display Networks

Showing inappropriate content on public display can have significant implications, with the potential to detract from an organisation's reputation and financial success. In current deployments, content selection is tightly-controlled with items typically coming from a single known source and so the risk of inappropriate content appearing at the display is low. In an open network with content integrated from multiple sources, the risk of an unknown content item not matching a display owner's expectations for the space is much greater.

Moderation techniques in other domains typically take one of the following forms: *pre-moderation* in which contributed items are placed in a queue to be checked and must be approved by a service manager to become visible to others; *post-moderation*, in which items become visible immediately but are then queued for moderation and may later be recalled by a service manager; *automated-moderation*, in which content is run through some sort of automatic check (e.g. a word filter) to determine whether or not it becomes visible; *reactive-moderation* in which the service provider relies on the community of service users to flag inappropriate content that is then removed.

Our evaluation work looking at display owner's attitudes to Tacita showed that most of our display owners felt that some moderation will be needed if viewer-contributed content becomes commonplace on public displays. However, it also showed that display owners were aware of the burden associated with content moderation – none of our suggested moderation techniques scored particularly highly and many of the owners cited the practicality of checking content as a key factor in their decision-making. Developing practical models for content moderation is therefore an important area for future work.

7.3.3 Digital Signage Analytics

Analytics measurements have become a valuable tool for understanding behaviour on the World Wide Web and on mobile devices. Such packages are critical to understanding and optimising content and applications, helping to identify the aspects that engage users and to build profiles of the kinds of content that attract different groups of individuals. Within the pervasive display domain similar tools could have a significant impact, enabling display owners to fine-tune content to meet viewers' needs and allowing content creators to ensure their content reaches those for whom it was intended.

Recent commercial systems have begun to explore the use of signage analytics with a focus on simple audience metrics to identify the age or gender of those in front of the display [Int]. However, a key focus of web analytics is the ability to be able to track 'click-throughs', i.e. actions that a user carries out as a result of viewing a specific piece of content. Making similar connections in the digital signage domain could involve, for example, identifying when a display viewer enters a particular coffee shop after seeing a menu or advertisement.

The multi-motivated nature of real-world actions poses a considerable challenge for digital signage analytics, and unlike web click-throughs, actions resulting from signage viewing are likely to occur some time after the viewer has moved away from a display. Identifying cause and effect for signage analytics is therefore an area of great challenge.

The varied nature of display deployments and usage patterns has the potential to generate vast quantities of data about the viewer (e.g. age, gender, speed and direction of travel), the display hardware (e.g. form factor, orientation, physical positioning) and the social and physical environment (e.g. cafe, office, or library; temperature, time, air pollution). Identifying patterns and representing this data in a meaningful way is another area of challenge within this research domain.

The application store described in this thesis provides a unique platform for distributing applications in open display networks. The emergence of similar platforms in the mobile device domain were one of the key drivers for the emergence of mobile analytics tools and provided an ideal host for such systems. In a similar way, we hope that Mercury and other content distribution tools will promote a new strand of research into digital signage analytics.

7.3.4 Consistent Interaction Patterns for Large Display Networks

The topic of interaction with pervasive displays has yielded a huge range of technologies and toolkits. Mobile-phone based interaction has been one popular area for research (e.g. [BGS+11, BJB09]), gesture-based methods are another common approach (e.g. [HRD11]), and recent research has also explored the use of eye-tracking (e.g. [SHZF10, VBG13]); within real-world deployments, touch-screens are undoubtedly the most popular hardware for supporting viewer interaction. Despite a significant body of research around interaction, we are yet to see consensus on the mechanisms by which interaction should be supported on pervasive displays. Current display deployments provide no common way to indicate if a display is interactive, which interaction technologies might be supported, or provide more general interaction structures in terms of consistent layouts and menus.

For viewer appropriation of pervasive displays to become commonplace in real-world deployments, viewers must be supported in the development of mental models about how to interact with a system once they have appropriated it. This is particularly important for active appropriation in which a viewer is explicitly using the display to achieve some specific purpose. However, the introduction of an application store to serve as a distribution mechanism and open up content production for pervasive display networks has the potential to increase the diversity of interaction approaches, potentially causing significant problems for appropriating viewers.

Exploring technologies and toolchains for creating consistent interaction patterns is therefore an important area for future work.

7.3.5 Understanding the Value of Appropriation in Real-World Deployments

The work in this thesis was partially motivated by recent research that indicates that viewers typically fail to attend to the displays in their environment due to a perception that the content is unlikely to be relevant [MWE+09]. Validating that appropriation has an impact on viewer engagement can only be established through real-world studies based on a long-term deployment.

Our evaluation studies suggest that display owners would be open to the support for viewer appropriation that our architecture provides, and that use cases for display viewers exist. However, this evidence is based on user surveys, interviews and focus groups in which participants reported their expectations. Self-reporting offers useful insight but can differ from real-world behaviour.

Studying use of the architecture in a real-world deployment would provide evidence to indicate whether supporting viewer appropriation can improve content relevance and increase engagement. For example, replicating prior studies (such as those by Huang et al. [HKB08] and Müller et al. [MWE+09]) in a deployment with appropriation support would allow comparison of viewing durations to establish if appropriation does address display blindness. Study of a real-world deployment could also identify usage patterns to establish the applications to which appropriation is well-suited. Similarly, data from a deployment would provide more substantial evidence regarding the factors that impact display owners' acceptance of viewer appropriation of their displays.

We therefore identify a need for future work that provides longitudinal studies exploring the value of appropriation in real-world deployments.

7.4 Closing Remarks

Pervasive digital displays are commonplace in today's world. Despite the potential for innovative engagement models, such displays often follow the traditional broadcasting models introduced with other media types (e.g. print, radio, television). Personalisation and appropriation allow the viewer of a display to influence, or have complete control over, the content shown on the displays in their environment, improving content relevance and increasing engagement – in short, allowing pervasive displays to realise their potential as a unique communication platform. Moving away from small-scale, closed, systems with traditional broadcasting models towards large, open, networks of pervasive displays which users can appropriate for their own purposes poses significant challenges, making openness in pervasive display systems an important research topic.

In this thesis we have explored the design space for user appropriation in such open pervasive display systems. We have presented an architecture that enables networks of public displays to access content from a wide range of sources. We have described a component-based software system for content playback allowing flexibility for new media types, network protocols and contextual data. We have detailed a system for allowing display viewers to make appropriation requests of the displays they encounter, using existing trust relationships with applications and services to prevent the disclosure of individual behaviour and preferences to display infrastructure. Finally, we have described an application store for content distribution and selection that builds upon lessons learnt in the personal mobile device domain, and encourages a separation of roles between those who create content and those who consume it. Together these systems form an integrated architecture for appropriation support in open display networks – the architecture was evaluated through a combination of quantitative and qualitative measurement.

It is hoped that the work presented in this thesis provides a foundation for the widespread deployment of pervasive display networks that are open to content from a range of sources, controlled by the display viewers themselves. We envisage future work in this space developing the work presented, particularly with regard to application stores for distribution and selection of display content. We hope that deployment of such systems in public and commercial environments will allow users to gain value from existing pervasive displays, improving relevance and engagement.

Appendix A

Content Descriptor Set (CDS) Schema Definition

```
<?xml version="1.0"?>


<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">
  <xs:element name="content-set" type="cdsType"/>


  <xs:complexType name="_withpreambleType">
    <xs:sequence>
      <xs:element name="auth" type="authType" minOccurs="0"
       maxOccurs="1"></xs:element>
      <xs:element name="feedback" minOccurs="0" maxOccurs="1" type
       ="feedbackType"></xs:element>
      <xs:element name="constraints" type="constraintsType"
       minOccurs="0" maxOccurs="1"></xs:element>
    </xs:sequence>
    <xs:attribute name="type" type="typeType" default="remote"/>
  </xs:complexType>

  <xs:complexType name="authType">
    <xs:attribute name="handler" type="authHandlerType" use="
      required" />
  </xs:complexType>

  <xs:simpleType name="authHandlerType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="none"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="feedbackType">
    <xs:complexContent>
      <xs:restriction base="xs:anyType"/>
    </xs:complexContent>
  </xs:complexType>

  <xs:simpleType name="typeType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Case insensitive: 'inline', 'remote'.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
          <xs:pattern value="((I|i)(N|n)(L|l)(I|i)(N|n)(E|e))|((R|
            r)(E|e)(M|m)(O|o)(T|t)(E|e))"/>
      <xs:enumeration value="inline"/>
      <xs:enumeration value="remote"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="cdsType">
    <xs:complexContent>
      <xs:extension base="_withpreambleType">
        <xs:choice>
          <xs:sequence>
            <xs:element name="requires-file" type="requiresType"
             minOccurs="0" maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation xml:lang="en">
                  If this content-set is remote then at least one
                   requires-file is needed.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="content-set" type="cdsType"
             minOccurs="0" maxOccurs="unbounded"></xs:element>
            <xs:element name="content-item" type="itemType"
             minOccurs="0" maxOccurs="unbounded"></xs:element>
          </xs:sequence>
        </xs:choice>
        <xs:attribute name="name" type="xs:string"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

```
<xs:complexType name="itemType">
  <xs:complexContent>
    <xs:extension base="_withpreambleType">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="requires-file" type="requiresType">
          <xs:annotation>
            <xs:documentation xml:lang="en">
                If this content-item is remote then at least one
                  requires-file is needed.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="content-type" type="xs:string" use="
       required" />
      <xs:attribute name="size" type="xs:string" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="constraintsType">
  <xs:sequence minOccurs="0" maxOccurs="1">
    <xs:element name="scheduling-constraints">
      <xs:complexType>
        <xs:all>
          <xs:element name="playback" minOccurs="0" maxOccurs="1
           ">
            <xs:complexType>
              <xs:attribute name="order" type="orderType"
               default="random" />
              <xs:attribute name="avoid-context-switch" type="
               xs:boolean" default="false" />
              <xs:attribute name="ratio" type="xs:string" />
            </xs:complexType>
          </xs:element>
          <xs:element name="time" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="between">
                  <xs:complexType>
                    <xs:attribute name="start" type="xs:time"
                     default="00:00:00" />
                    <xs:attribute name="end" type="xs:time"
                     default="23:59:59" />
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="day-of-week" minOccurs="0" maxOccurs
           ="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="between">
                  <xs:complexType>
                    <xs:attribute name="start" type="dowType"
                     use="required" />
```

```
                <xs:attribute name="end" type="dowType" use=
                 "required" />
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="date" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="between">
              <xs:complexType>
                <xs:attribute name="start" type="xs:date"
                 use="required" />
                <xs:attribute name="end" type="xs:date" use=
                 "required" />
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="priority" minOccurs="0" maxOccurs="1
       ">
        <xs:complexType>
          <xs:attribute name="level" type="priorityType"
           default="medium" />
        </xs:complexType>
      </xs:element>
    </xs:all>
  </xs:complexType>
```

```
      </xs:element>
      <xs:element name="input-constraints" minOccurs="0" maxOccurs
       ="1">
        <xs:complexType>
          <xs:complexContent>
            <xs:restriction base="xs:anyType"/>
          </xs:complexContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="output-constraints" minOccurs="0"
       maxOccurs="1">
        <xs:complexType>
          <xs:complexContent>
            <xs:restriction base="xs:anyType"/>
          </xs:complexContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="requiresType">
    <xs:sequence>
      <xs:element name="hashes" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="hash" minOccurs="0" maxOccurs="
             unbounded">
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:string">
                    <xs:attribute name="type" type="xs:string"
                     use="required"/>
                  </xs:extension>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="sources" minOccurs="1" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="uri" minOccurs="1" maxOccurs="
             unbounded">
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:anyURI">
                    <xs:attribute name="refresh" type="xs:string"/
                     >                  </xs:extension>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="dowType">
    <xs:annotation>
```

```
      <xs:documentation xml:lang="en">
        Case insensitive: 'mon[day]', 'tue[sday]', 'wed[nesday]',
          'thu[rsday]', 'fri[day]', 'sat[urday]', 'sun[day]'
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="((M|m)(O|o)(N|n)((D|d)(A|a)(Y|y))?)|((T|t
        )(U|u)(E|e)((S|s)(D|d)(A|a)(Y|y))?)|((W|w)(E|e)(D|d)((N|n)(
        E|e)(S|s)(D|d)(A|a)(Y|y))?)|((T|t)(H|h)(U|u)((R|r)(S|s)(D|d
        )(A|a)(Y|y))?)|((F|f)(R|r)(I|i)((D|d)(A|a)(Y|y))?)|((S|s)(A
        |a)(T|t)((U|u)(R|r)(D|d)(A|a)(Y|y))?)|((S|s)(U|u)(N|n)((D|d
        )(A|a)(Y|y))?)"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="orderType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Case insensitive: '[in]order', 'rand[om]', 'reverse[order]
          '
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:pattern value="(((I|i)(N|n))?(O|o)(R|r)(D|d)(E|e)(R|
          r))|((R|r)(A|a)(N|n)(D|d)((O|o)(M|m))?)|((R|r)(E|e)(V|v
          )(E|e)(R|r)(S|s)(E|e)((O|o)(R|r)(D|d)(E|e)(R|r))?)"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="priorityType">
```

```
      <xs:annotation>
        <xs:documentation xml:lang="en">
          A priority type field may contain either a string or
           integer value.
          Acceptable string values (case-insensitive): 'lowest', '
           low', 'medium',
          'high', 'highest'. A value of 0 is equivalent to 'lowest'
           priority and
          a value of 4 is equivalent to 'highest' priority.
        </xs:documentation>
      </xs:annotation>
      <xs:union>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:pattern value="((L|l)(O|o)(W|w)(E|e)(S|s)(T|t))|((L|
             l)(O|o)(W|w))|((M|m)(E|e)(D|d)(I|i)(U|u)(M|m))|((H|h)(I
             |i)(G|g)(H|h))|((H|h)(I|i)(G|g)(H|h)(E|e)(S|s)(T|t))"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:nonNegativeInteger">
              <xs:maxInclusive value="4" />
          </xs:restriction>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>

  </xs:schema>
```

Listing A.1: XML Schema (XSD) for our Content Descriptor Set format

References

[ABK+09] Florian Alt, Moritz Balz, Stefanie Kristes, Alireza Sahami Shirazi, Julian Mennenöh, Albrecht Schmidt, Hendrik Schröder, and Michael Goedicke. Adaptive user profiles in pervasive advertising environments. In *Proceeedings of the European Conference on Ambient Intelligence*, AmI '09, pages 276–286, Berlin, Heidelberg, 2009. Springer-Verlag. 59, 90, 91

[ACH+01] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper. Implementing a sentient computing system. *Computer*, 34(8):50–56, August 2001. 192

[Ads14] Adspace Networks, Inc. Adspace digital mall network. `http://www.adspacenetworks.com/` [Last accessed: April 2014], 2014. 60

[Aga03] Stefan Agamanolis. Designing displays for human connectedness. In Kenton O'Hara, Mark Perry, Elizabeth Churchill, and Daniel M. Russel, editors, *Public and Situated Displays – Social and Interactional Aspects of Shared Display Technologies*, pages 309–334. Kluwer, 2003. 76

[AKB+11] Florian Alt, Thomas Kubitza, Dominik Bial, Firas Zaidan, Markus Ortel, Björn Zurmaar, Tim Lewen, Alireza Sahami Shirazi, and Albrecht Schmidt. Digifieds: Insights into deploying digital public notice areas in the wild. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, MUM '11, pages 165–174, New York, NY, USA, December 2011. ACM. 51, 90, 104, 238

[Ale13] Alex Gaynor and others. django-filter. https://django-filter.readthedocs.org [Last accessed: October 2013], 2013. 215

[Ama12] Amazon Web Services LLC. Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/ [Last accessed: July 2012], 2012. 117

[AME+11] Florian Alt, Nemanja Memarovic, Ivan Elhart, Dominik Bial, Albrecht Schmidt, Marc Langheinrich, Gunnar Harboe, Elaine M. Huang, and Marcello P. Scipioni. Designing shared public display networks – implications from today's paper-based notice areas. In *Proceedings of the 9th International Conference on Pervasive Computing*, Pervasive '11, Berlin, Heidelberg, June 2011. Springer-Verlag. 37

[AMS12] Florian Alt, Jörg Müller, and Albrecht Schmidt. Advertising on public display networks. *Computer, IEEE*, 45(5):50–56, May 2012. 58, 102, 140, 153

[AOSP13] The Android Open Source Project. Locationmanager | android developers. http://developer.android.com/reference/android/location/LocationManager.html [Last accessed: November 2013], 2013. 226

[App13] Apple Inc. Core location framework reference. https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CoreLocation_Framework/_index.html [Last accessed: November 2013], 2013. 226

[App14a] AppBrain. Number of available Android applications - AppBrain. http://www.appbrain.com/stats/number-of-android-apps [Last accessed: May 2014], May 2014. 288

[App14b] Apple Inc. App Store sales top $10 billion in 2013. http://www.apple.com/pr/library/2014/01/

07App-Store-Sales-Top-10-Billion-in-2013.html [Last accessed: May 2014], January 2014. 288

[App14c] ApplianceStudio. Appliancestudio: Printsign. http://www.ambientweb.co.uk/sectors/smartsigns/printsign.htm [Last accessed: April 2014], 2014. 61

[Art12] Artichoke. The Telectroscope by Paul St George. 22 May – 15 June 2008. London | New York. http://www.talktalk.co.uk/telectroscope/home.php [Last accessed: October 2012], 2008–2012. 19

[AS03] Stavros Antifakos and Bernt Schiele. Laughinglily: Using a flower as a real world information display. In *Proceedings of the 5th International Conference on Ubiquitous Computing*, Ubicomp '03, October 2003. 22, 90

[AS12] Florian Alt and Stefan Schneegass. A conceptual architecture for pervasive advertising in public display networks. In *Proceedings of the 3rd Workshop on Infrastructure and Design Challenges of Coupled Display Visual Interfaces*, PPD '12, May 2012. 57

[ASKS13] Florian Alt, Alireza Sahami Shirazi, Thomas Kubitza, and Albrecht Schmidt. Interaction techniques for creating and exchanging content with public displays. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '13, pages 1709–1718, New York, NY, USA, April 2013. ACM. 51, 238

[ATT99] AT&T Laboratories Cambridge. VNC – virtual network computing. http://www.cl.cam.ac.uk/research/dtg/attarchive/vnc/index.html [Last accessed: December 2012], 1999. 82, 191

[BAB$^+$07] Sebastian Boring, Manuela Altendorfer, Gregor Broll, Otmar Hilliges, and Andreas Butz. Shoot & copy: Phonecam-based information transfer from public displays onto mobile phones. In *Proceedings of the 4th* *international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, Mobility '07, pages 24–31, New York, NY, USA, 2007. ACM. 51

[Bal02] Philip Ball. Tv on a t-shirt. http://www.nature.com/news/1998/020520/full/news020520-4.html [Last accessed: November 2012], May 2002. 21

[Bal05] Rafael Ballagas. Sweep and point & shoot: Phonecam-based interactions for large public displays. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '05, pages 1200–1203, New York, NY, USA, April 2005. ACM. 46, 90

[BAM$^+$11] Gilbert Beyer, Florian Alt, Jörg Müller, Albrecht Schmidt, Ivo Haulsen, Stefan Klose, Karten Isakovic, and Manuel Schiewe. Audience behavior around large interactive cylindrical screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, New York, NY, USA, 2011. ACM. 69, 70

[BB13] BBC Connected Studios and BBC News Labs. #newsHACK. http://newshack.co.uk/ [Last accessed: November 2013], 2013. 239

[BBB$^+$10] Sebastian Boring, Dominikus Baur, Andreas Butz, Sean Gustafson, and Patrick Baudisch. Touch projector: Mobile interaction through video. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2287–2296, New York, NY, USA, 2010. ACM. 47

[BBC12] BBC. BBC - big screens. http://www.bbc.co.uk/bigscreens/ [Last accessed: December 2012], 2012. 38, 90

[BBC13a] BBC. BBC - homepage. http://www.bbc.co.uk/ [Last accessed: January 2013], 2013. 38

[BBC13b] BBC. Bbc developer portal i/o docs. http://bbc.mashery.com/io-docs [Last accessed: November 2013], 2013. 228

[BEA14] BEABLOO, S.L. Beabloo Digital Marketing and Digital Signage Solutions. http://www.beabloo.com [Last accessed: May 2014], 2014. 61

[BGS+11] Gregor Broll, Roman Graebsch, Maximilian Scherr, Sebastian Boring, Paul Holleis, and Matthias Wagner. Touch to play – exploring touch-based mobile interaction with public displays. In *Proceedings of the 3rd International Workshop on Near Field Communication*, NFC '11, pages 15–20, 2011. 44, 104, 308

[BHI93] Sara A. Bly, Steve R. Harrison, and Susan Irwin. Media spaces: Bringing people together in a video, audio, and computing environment. *Communications of the ACM*, 36(1):28–46, January 1993. 19

[BHS06] Jakob E. Bardram, Thomas R. Hansen, and Mads Soegaard. Aware-media: a shared interactive display supporting social, temporal, and spatial awareness in surgery. In *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, CSCW '06, pages 109–118, New York, NY, USA, 2006. ACM. 33, 90

[BIF+04] Harry Brignull, Shahram Izadi, Geraldine Fitzpatrick, Yvonne Rogers, and Tom Rodden. The introduction of a shared interactive surface into a communal space. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 49–58, New York, NY, USA, 2004. ACM. 35, 84, 104, 290

[BJ04] Russell Beale and Matthew Jones. Integrating situated interaction with mobile awareness. In *Proceedings of MLEARN*, 2004. 29

[BJB09] Sebastian Boring, Marko Jurmu, and Andreas Butz. Scroll, tilt or move it: Using mobile phones to continuously control pointers on large public displays. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7*, OZCHI '09, pages 161–168, New York, NY, USA, November 2009. ACM. 49, 90, 308

[BKN05a] Ricardo A. Baratto, Leonard N. Kim, and Jason Nieh. THINC: A virtual display architecture for thin-client computing. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, SOSP '05, pages 277–290, New York, NY, USA, 2005. ACM. 81

[BKN05b] Stefan Berger, Rick Kjelsen, and Chandra Narayanaswami. Using symbiotic displays to view sensitive information in public. In *Proceedings of the 3rd International Conference on Pervasive Computing and Communications*, 2005. 55, 191

[BKW00] Susanne Boll, Wolfgang Klas, and Utz Westermann. Multimedia document models: Sealed fate or setting out for new shores? *Multimedia Tools Appl.*, 11(3):267–279, August 2000. 166

[Bla13] BlackBerry Limited. BlackBerry world - BlackBerry developer. `http://developer.blackberry.com/blackberryworld/` [Last accessed: May 2014], July 2013. 288

[BM98] Marc Böhlen and Michael Mateas. Office Plant #1: Intimate space and contemplative entertainment. *Leonardo*, 31(5):345–348, 1998. 22, 90

[BMMR96] Richard Borovoy, M. McDonald, Fred Martin, and Mitchel Resnick. Things that blink: Computationally augmented name tags. *IBM Systems Journal*, 35(3-4):488–495, September 1996. 20, 33

[BMRS98] Richard Borovoy, Fred Martin, Mitchel Resnick, and Brian Silverman. Groupwear: Nametags that tell about relationships. In *Conference Summary on Human Factors in Computing Systems*, CHI '98, pages 329–330, New York, NY, USA, April 1998. ACM. 20, 33

[BMV$^+$98] Richard Borovoy, Fred Martin, Sunil Vemuri, Mitchel Resnick, Brian Silverman, and Chris Hancock. Meme tags and community mirrors: Moving from conferences to collaboration. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, CSCW '98, pages 159–168, New York, NY, USA, 1998. ACM. 20, 33

[Bor02] Richard Borovoy. *Folk Computing: Designing Technology to Support Face-to-Face Community Building*. PhD thesis, Massachusetts Institute of Technology, 2002. 20, 33

[BR03] Harry Brignull and Yvonne Rogers. Enticing people to interact with large public displays in public spaces. In *Proceedings of the IFIP International Conference on Human-Computer Interaction*, INTERACT '03, pages 17–24. IOS Press, September 2003. 34, 63, 64, 70, 76, 90

[BRH94] F. Bennett, T. Richardson, and A. Harter. Teleporting – making applications mobile. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, pages 82–84, Washington, DC, USA, 1994. IEEE Computer Society. 82

[BRHW11] Gregor Broll, Wolfgang Reithmeier, Paul Holleis, and Matthias Wagner. Design and evaluation of techniques for mobile interaction with dynamic NFC-displays. In *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '11, pages 205–212, New York, NY, USA, 2011. ACM. 44

[Bro14] BroadSign International, LLC. Digital signage software solutions - broadsign. `http://broadsign.com/` [Last accessed: April 2014], 2014. 61

[Bui13] Buildbot Team Members. Mercurial scm. `http://trac.buildbot.net/` [Last accessed: October 2013], 2013. 196, 216

[CAN08] Ben Congleton, Mark S. Ackerman, and Mark W. Newman. The ProD framework for proactive displays. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, pages 221–230, New York, NY, USA, 2008. ACM. 33, 86, 96

[CCF95] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. 3-dimensional pliable surfaces: For the effective presentation of visual information. In *Proceedings of the 8th Annual ACM Symposium on User Interface Software and Technology*, UIST '95, pages 217–226, New York, NY, USA, 1995. ACM. 30

[CDF$^+$05a] Keith Cheverst, Alan Dix, Daniel Fitton, Christian Kray, Mark Rouncefield, Corina Sas, George Saslis-Lagoudakis, and Jennifer G. Sheridan. Exploring Bluetooth based mobile phone interaction with the Hermes photo display. In *Proceedings of the 7th International Conference on Human Computer Interaction with Mobile Devices & Services*, MobileHCI '05, pages 47–54, New York, NY, USA, 2005. ACM. 32, 42, 90

[CDF$^+$05b] Keith Cheverst, Alan Dix, Daniel Fitton, Christian Kray, Mark Rouncefield, George Saslis-Lagoudakis, and Jennifer G. Sheridan. Exploring mobile phone interaction with situated displays. In *Proceedings of Pervasive Mobile Interaction Devices (PERMID) 2005 at the 3rd International Conference on Pervasive Computing (Pervasive '05)*, 2005. 32, 42

[CDF$^+$07] Keith Cheverst, Alan Dix, Daniel Fitton, Mark Rouncefield, and Connor Graham. Exploring awareness related messaging through two situated-display-based systems. *Human-Computer Interaction*, 22:173–220, 2007. 41

[CDFC13] Sarah Clinch, Nigel Davies, Adrian Friday, and Graham Clinch. Yarely – a software player for open pervasive display networks. In *Proceedings of the 2013 International Symposium on Pervasive Displays*, PerDis '13, New York, NY, USA, 2013. ACM. i, 149, 195, 240

[CDFE11a] Sarah Clinch, Nigel Davies, Adrian Friday, and Christos Efstratiou. Reflections on the long-term use of an experimental public display system. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, Ubicomp '11, pages 133–142, 2011. i, 94, 138, 182

[CDFE11b] Sarah Clinch, Nigel Davies, Adrian Friday, and Christos Efstratiou. Reflections on the long-term use of an experimental public display system. In *Poster at Ubicomp '11*, 2011. 94

[CDKS12] Sarah Clinch, Nigel Davies, Thomas Kubitza, and Albrecht Schmidt. Designing application stores for public display networks. In *Pro-*

ceedings of the 2012 International Symposium on Pervasive Displays, PerDis '12, New York, NY, USA, 2012. ACM. i, ii, 149, 173

[CFD03] Keith Cheverst, Daniel Fitton, and Alan Dix. Exploring the evolution of office door displays. In Kenton O'Hara, Mark Perry, Elizabeth Churchill, and Daniel M. Russel, editors, *Public and Situated Displays – Social and Interactional Aspects of Shared Display Technologies*, pages 141–169. Kluwer, 2003. 28, 42

[CFDR02] Keith Cheverst, Daniel Fitton, Alan Dix, and Mark Rouncefield. Exploring situated interaction with ubiquitous office door displays. In *Public, Community and Situated Displays (Workshop at CSCW 2002)*, New Orleans, LA, USA, 2002. 28, 42

[CGNL04] Elizabeth F. Churchill, Andreas Girgensohn, Les Nelson, and Alison Lee. Blending digital and physical spaces for ubiquitous community participation. *Communications of the ACM*, 47(2):38–44, February 2004. 69

[CHF$^+$12] Sarah Clinch, Jan Harkes, Adrian Friday, Nigel Davies, and Mahadev Satyanarayanan. How close is close enough? understanding the role of cloudlets in supporting display appropriation by mobile users. In *Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communication*, PerCom '12, pages 122–127, 2012. i, 94, 127

[Chr13] Tom Christie. Django rest framework: Awesome web-browsable web apis. `http://django-rest-framework.org/` [Last accessed: October 2013], 2011–2013. 215

[CKDL12a] Sarah Clinch, Thomas Kubitza, Nigel Davies, and Marc Langheinrich. Demo: Using mobile devices to personalize pervasive displays. In *Demo. at Mobisys '12*, 2012. i, 194, 239

[CKDL12b] Sarah Clinch, Thomas Kubitza, Nigel Davies, and Marc Langheinrich. Using mobile devices to personalize pervasive displays. In *Demo. at Digital Futures '12*, 2012. i, 195, 239

[Cli13] Sarah Clinch. Smartphones and pervasive public displays. *Pervasive Computing, IEEE*, 12(1):92–95, 2013. i, 18

[CLMT12] Kelvin Cheng, Jane Li, and Christian Müller-Tomfelde. Supporting interaction and collaboration on large displays using tablet devices. In *Proceedings of the 2012 Workshop on Infrastructure and Design Challenges of Couple Display Visual Interfaces at the International Working Conference on Advanced Visual Interfaces (AVI 2012)*, PPD '12, pages 14–17, May 2012. 56, 90

[CLS$^+$05] Andrea Chew, Vincent Leclerc, Sajid Sadi, Aaron Tang, and Hiroshi Ishii. SPARKS. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '05, pages 1276–1279, New York, NY, USA, 2005. ACM. 36

[CMB08] Xiang Cao, Michael Massimi, and Ravin Balakrishnan. Flashlight Jigsaw: an exploratory study of an ad-hoc multi-player game on public displays. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, CSCW '08, pages 77–86, New York, NY, USA, 2008. ACM. 48, 69, 74, 75, 77, 78, 91, 104, 265

[CMKK11] Cédric Cochrane, Ludivine Meunier, Fern M. Kelly, and Valdan Koncar. Flexible displays for smart clothing : Part I — overview. *Indian Journal of Fibre and Textile Research*, 36:422–428, December 2011. 21

[CNDG03] Elizabeth F. Churchill, Les Nelson, Laurent Denoue, and Andreas Girgensohn. The plasma poster network: Posting multimedia content in public places. In *Proceedings of the 9th IFIP TC13 International Conference on Human-Computer Interaction*, INTERACT '03. IOS Press, September 2003. 31, 90, 104

[Con07] Ben Congleton. A 'visual commons' framework for public displays. `http://hdl.handle.net/10535/3816` [Last accessed: January 2013], 2007. 33, 85, 91, 163

[CTF+12] Keith Cheverst, Faisal Taher, Matthew Fisher, Daniel Fitton, and Nick Taylor. The design, deployment and evaluation of situated display-based systems to support coordination and community. In Antonio Krüger and Tsvi Kuflik, editors, *Ubiquitous Display Environments*, Cognitive Technologies, pages 105–124. Springer-Verlag, Berlin, Heidelberg, 2012. 28, 32, 39, 42

[CTR+08] Keith Cheverst, Nick Taylor, Mark Rouncefield, Areti Galani, and Christian Kray. The challenge of evaluating situated display based technology interventions designed to foster a sense of community. In *Proceedings of Ubiquitous Systems Evaluation Workshop at the 10th ACM International Conference on Ubiquitous Computing (Ubicomp '08)*, 2008. 39

[DCAss] Nigel Davies, Sarah Clinch, and Florian Alt. *Pervasive Displays: Understanding the Future of Digital Signage*. Synthesis Lectures on Computer Science. Morgan & Claypool Publishers, In Press. i, 18

[Des14] Destination Media. Gas station TV. www.gstv.com [Last accessed: May 2014], 2014. 60

[DFCS10] Nigel Davies, Adrian Friday, Sarah Clinch, and Albrecht Schmidt. Challenges in developing an app store for public displays – a position paper. In *Proceedings of "Research in the Large" Workshop at Ubicomp '10*, 2010. i, ii, 149

[DFN+09a] Nigel Davies, Adrian Friday, Peter Newman, Sarah Rutlidge, and Oliver Storz. Using bluetooth device names to support interaction in smart environments. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, MobiSys '09, 2009. i, 94, 106, 108, 114, 163

[DFN+09b] Nigel Davies, Adrian Friday, Peter Newman, Sarah Rutlidge, and Oliver Storz. Using bluetooth device names to support interaction in smart environments. In *Demo at HotMobile 2009*, 2009. 94

[DGT+13] Artem Dementyev, Jeremy Gummeson, Derek Thrasher, Aaron Parks, Deepak Ganesan, Joshua R. Smith, and Alanson P. Sample. Wirelessly powered bistable display tags. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 383–386, New York, NY, USA, 2013. ACM. 44

[Dig12] Digital Signage Networks India Pvt. Ltd. DSN – digital signage networks. `http://app.dsnglobal.com/` [Last accessed: September 2012], 2012. 2, 37, 171

[Dja13a] Django Software Foundation. Django – the web framework for perfectionists with deadlines. `https://www.djangoproject.com/` [Last accessed: October 2013], 2005–2013. 115, 212

[Dja13b] Django Software Foundation. Writing your first django app, part 1. `https://www.djangoproject.com/` [Last accessed: November 2013], 2005–2013. 216

[DLC+14] Nigel Davies, Marc Langheinrich, Sarah Clinch, Adrian Friday, Ivan Elhart, Thomas Kubitza, and Bholanathsingh Surajbali. Personalisation and privacy in future pervasive display networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, New York, NY, USA, 2014. ACM. i, 149, 189, 195, 241, 258, 261, 262

[DLJS12] Nigel Davies, Marc Langheinrich, Rui José, and Albrecht Schmidt. Open display networks: A communications medium for the 21st century. *Computer, IEEE*, 45(5):58–64, May 2012. xiv, 2, 3, 166, 190, 288, 299

[DNM+10] Charles Dennis, Andrew Newman, Richard Michon, J. Josko Brakus, and Len Tiu Wright. The mediating effects of perception and emotion: Digital signage in mall atmospherics. *Journal of Retailing and Consumer Services*, 17(3):205–215, May 2010. 265

[DT09] David Dearman and Khai Truong. BlueTone: A framework for interacting with public displays using dual-tone multi-frequency through bluetooth. In *Proceedings of the 11th ACM International Conference on Ubiquitous Computing*, Ubicomp '09, September 2009. 43

[DWI98] Andrew Dahley, Craig Wisneski, and Hiroshi Ishii. Water Lamp and Pinwheels: Ambient projection of digital information into architectural space. In *Conference Summary on Human Factors in Computing Systems*, CHI '98, pages 269–270, New York, NY, USA, April 1998. ACM. 22

[DYN14] DYNAMAX TECHNICAL SERVICES Ltd. About POV^{NG}. http://www.dynamaxworld.com/products/about-povng/ [Last accessed: April 2014], 2014. 61

[Eas04] Douglas Easterly. Bio-Fi: Inverse biotelemetry projects. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, MULTIMEDIA '04, pages 182–183, New York, NY, USA, October 2004. ACM. 22

[EBF$^+$08] Aiman Erbad, Michael Blackstock, Adrian Friday, Rodger Lea, and Jalal Al-Muhtadi. Magic broker: A middleware toolkit for interactive public displays. In *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, PerCom '08, pages 509–514, Washington, DC, USA, 2008. IEEE Computer Society. 41, 90

[ET08] James R. Eagan and John T.Stasko. The buzz: Supporting user tailorability in awareness applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1729–1738, New York, NY, USA, 2008. ACM. 87

[Fac13] Facebook. Tornado web server – tornado 3.1.1 documentation. http://www.tornadoweb.org/en/stable/ [Last accessed: November 2013], 2013. 196, 228

[Far01] Stephen Farrell. Social and informational proxies in a fishtank. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '01, pages 365–366, New York, NY, USA, March 2001. ACM. 25

[Fas04] Adam M. Fass. MessyBoard: Lowering the cost of communication and making it more enjoyable. In *Doctorial Symposium at the Seventeenth Annual ACM Symposium on User Interface Software and Technology (UIST 2004)*, 2004. 31

[FB99] Jennica Falk and Staffan Björk. The BubbleBadge: A wearable public display. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '99, pages 318–319, New York, NY, USA, May 1999. ACM. 21, 90

[FCFD04] Daniel Fitton, Keith Cheverst, Joe Finney, and Alan Dix. Supporting interaction with office door displays. In *Workshop on Multi-User and Ubiquitous User Interfaces 2004*, MU3I 2004, 2004. 28, 42

[FCK+04] Daniel Fitton, Keith Cheverst, Chris Kray, Alan Dix, Mark Rouncefield, and George Saslis-Lagoudakis. Rapid prototyping and user-centred design of interactive display-based systems. *Pervasive Computing, IEEE*, 2004. 41

[FDE12] Adrian Friday, Nigel Davies, and Christos Efstratiou. Reflections on long-term experiments with public displays. *Computer, IEEE*, 45(5):34–41, May 2012. 8, 27, 39, 82, 182, 234

[FFP02] Adam M. Fass, Jodi Forlizzi, and Randy Pausch. MessyDesk and MessyBoard: Two designs inspired by the goal of improving human memory. In *Proceedings of the 4th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, DIS '02, pages 303–311, New York, NY, USA, 2002. ACM. 31

[FKC90] Robert S. Fish, Robert E. Kraut, and Barbara L. Chalfonte. The VideoWindow system in informal communication. In *Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work*, CSCW '90, pages 1–11, New York, NY, USA, 1990. ACM. 19

[FMP+09] Shelly D. Farnham, Joseph F. McCarthy, Yagnesh Patel, Sameer Ahuja, Daniel Norman, William R. Hazlewood, and Josh Lind. Measuring the impact of third place attachment on the adoption of a place-based community technology. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 2153–2156, New York, NY, USA, 2009. ACM. 36, 90

[Fog13a] Fog Creek Software, Inc. Kiln - version control and code review software from fog creek software. http://www.fogcreek.com/kiln/ [Last accessed: October 2013], 2000–2013. 196, 216

[Fog13b] Fog Creek Software, Inc. Bug tracking, done right | fogbugz from fog creek software. http://www.fogcreek.com/fogbugz/ [Last accessed: October 2013], 2000-2013. 196, 216

[FP12] FH-Potsdam. Public displays. http://interface.fh-potsdam.de/publicdisplays/ [Last accessed: April 2013], 2012. 45

[FS03] Steven L. Franconeri and Daniel J. Simons. Moving and looming stimuli capture attention. *Perception & Psychophysics*, 65(7):999–1010, 2003. 69, 91

[FTLB08] Matthias Finke, Anthony Tang, Rock Leung, and Michael Blackstock. Lessons learned: game design for large public displays. In *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts*, DIMEA '08, pages 26–33, New York, NY, USA, 2008. ACM. 42, 65, 69, 74, 104

[FV02] Alois Ferscha and Simon Vogl. Pervasive web access via public communication walls. In *Proceedings of the First International Conference on Pervasive Computing*, Pervasive '02, pages 84–97, London, UK, UK, 2002. Springer-Verlag. 45, 84, 91

[FV10] Alois Ferscha and Simon Vogl. Wearable displays for everyone! *Pervasive Computing, IEEE*, 9(1):7–10, January–March 2010. 21

[FWDF96] Joe Finney, Stephen Wade, Nigel Davies, and Adrian Friday. Flump: The FLexible Ubiquitous Monitor Project. In *Cabernet Radicals Workshop*, May 1996. 27, 82, 83, 90, 91, 96, 97, 160, 163, 183, 186, 192

[GA86] George O. Goodman and Mark J. Abel. Collaboration research in SCL. In *Proceedings of the 1986 ACM Conference on Computer-supported Cooperative Work*, CSCW '86, pages 246–251, New York, NY, USA, 1986. ACM. 19

[Gal08] The Daily Galaxy. Helsinki's "city wall" -a collaborative social space. http://www.dailygalaxy.com/my_weblog/2008/11/helsinkis-city.html [Last accessed: February 2012], November 2008. 38

[GCFR05] Connor Graham, Keith Cheverst, Dan Fitton, and Mark Rouncefield. "how do you turn a duck into a soul singer? put it in the microwave until its bill withers": Some social features of a simple technology. In *International Forum "Less is More – Simple Computing in an Age of Complexity"*. Microsoft Research, 2005. 41

[GCR05] Connor Graham, Keith Cheverst, and Mark Rouncefield. Technology for the humdrum: trajectories, interactional needs and a care setting. In *Proceedings of the 17th Australia Conference on Computer-Human Interaction: Considerations for Today and the Future*, OzCHI '05, pages 1–10, Narrabundah, Australia, Australia, November 2005. Computer-Human Interaction Special Interest Group (CHISIG) of Australia. 41

[GDL12] Sven Gehring, Florian Daiber, and Christian Lander. Towards universal, direct remote interaction with distant public displays. In *Proceedings of the 2012 Workshop on Infrastructure and Design Challenges of Couple Display Visual Interfaces at the International Working Conference on Advanced Visual Interfaces (AVI 2012)*, PPD '12, pages 14–17, May 2012. 47

[GFH+13] Jorge Goncalves, Denzil Ferreira, Simo Hosio, Yong Liu, Jakob Rogstadius, Hannu Kukka, and Vassilis Kostakos. Crowdsourcing on the spot: Altruistic use of public displays, f feasibility, performance, and behaviours. In *Proceedings of the 2013 ACM International Joint*

Conference on Pervasive and Ubiquitous Computing, UbiComp '13, pages 753–762, New York, NY, USA, 2013. ACM. 68, 71, 75, 76

[GHtXP14] Daniel Garner, Alex Harrington, and the Xibo Project. Xibo - digital signage. `http://xibo.org.uk/` [Last accessed: May 2014], 2006-2014. 61, 161

[GMRS03] Antonietta Grasso, Martin Muehlenbrock, Frederic Roull, and Dave Snowdon. Supporting communities of practice with large screen displays. In Kenton O'Hara, Mark Perry, Elizabeth Churchill, and Daniel M. Russel, editors, *Public and Situated Displays – Social and Interactional Aspects of Shared Display Technologies*, pages 261–282. Kluwer, 2003. 30, 31, 104

[Goo12] Google Inc. JSON developer's guide - news search (deprecated) – Google developers. `https://developers.google.com/news-search/v1/jsondevguide` [Last accessed: November 2013], 2012. 228

[Goo13a] Google Inc. The Google geocoding API. `https://developers.google.com/maps/documentation/geocoding/` [Last accessed: November 2013], 2013. 228

[Goo13b] Google Inc. The Google time zone API. `https://developers.google.com/maps/documentation/timezone/` [Last accessed: November 2013], 2013. 228

[Goo14] Google. Chrome remote desktop. `https://chrome.google.com/webstore/detail/chrome-remote-desktop/gbchcmhmhahfdphkhkmpfmihenigjmpp` [Last accessed: May 2014], April 2014. 80

[Gou03] Paula Gould. Textiles gain intelligence. *Materials Today*, 6:38–43, 2003. 21

[GP11] The GNOME Project. PyGObject – GLib/GObject/GIO Python bindings. https://wiki.gnome.org/PyGObject [Last accessed: October 2013], 2005–2011. 196

[GP13] The GNOME Project. gtk-vnc. https://wiki.gnome.org/gtk-vnc [Last accessed: November 2013], 2005–2013. 196

[GPP$^+$12] Giuseppe Ghiani, Fabio Paternó, Jussi Polet, Ville Antila, and Jani Mäntyjärvi. A context-dependent environment for web application migration. In *Proceedings of the 2012 Workshop on Infrastructure and Design Challenges of Couple Display Visual Interfaces at the International Working Conference on Advanced Visual Interfaces (AVI 2012)*, PPD '12, May 2012. 54

[GR80] Kit Galloway and Sherrie Rabinowitz. Hole-In-Space. http://www.ecafe.com/getty/HIS/ [Index from 7 August 2007 retrieved via the Internet Archive in October 2012 (http://web.archive.org/web/20100807022543/http://www.ecafe.com/getty/HIS/)], 1980. 18, 90

[GR01] Saul Greenberg and Michael Rounding. The notification collage: Posting information to public and personal displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 514–521, New York, NY, USA, March 2001. ACM. 30, 77, 160

[GR13] Daniel Greenfield and Audrey Roy. *Two Scoops of Django: Best Practices For Django 1.5.* CreateSpace Independent Publishing Platform, 2013. 216

[Gre99] Saul Greenberg. Designing computers as public artifacts. In *International Journal of Design Computing: Special Issue on Design Computing on the Net (DCNet '99)*, 1999. 30

[Gre13] Miriam Greis. Investigating conficts of interests between stakeholders of public display app stores. Diploma thesis, University of Stuttgart, 2013. 216

[GRK13] Brian E. Granger and Min Ragan-Kelley. PyZMQ documentation. http://zeromq.github.io/pyzmq/ [Last accessed: October 2013], 2013. 195

[Hai14] Haivision Network Video. CoolSign – Integrated Digital Signage & IP Video. http://www.haivision.com/products/digital-signage/coolsign [Last accessed: April 2014], 2014. 61

[HBL98] Stephanie Houde, Rachel Bellamy, and Laureen Leahy. In search of design principles for tools and practices to support communication within a learning community. *SIGCHI Bulletin*, 30(2):113–118, April 1998. 29

[HDM$^+$10] William R. Hazlewood, Nick Dalton, Paul Marshall, Yvonne Rogers, and Susanna Hertrich. Bricolage and consultation: Addressing new design challenges when building large-scale installations. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, DIS '10, pages 380–389, New York, NY, USA, 2010. ACM. 23, 40

[Hel13] Helsinki Institute for Information Technology. Citywall. http://citywall.org/ [Last accessed: February 2013], 2013. 38

[HHT99] Jeremy M. Heiner, Scott E. Hudson, and Kenichiro Tanaka. The information percolator: Ambient information display in a decorative object. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, UIST '99, pages 141–148, New York, NY, USA, 1999. ACM. 22, 90

[Hip13] Dwayne Richard Hipp. Sqlite home page. http://www.sqlite.org/ [Last accessed: November 2013], 2013. 228

[HKB08] Elaine M. Huang, Anna Koster, and Jan Borchers. Overcoming assumptions and uncovering practices: When does the public really look at public displays? In *Proceedings of the 6th International Conference on Pervasive Computing*, Pervasive '08, pages 228–243, Berlin, Heidelberg, May 2008. Springer-Verlag. 66, 91, 267, 309

[HKH$^+$04] David Holstius, John Kembel, Amy Hurst, Peng-Hui Wan, and Jodi Forlizzi. Infotropism: Living and robotic plants as interactive displays. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, DIS '04, pages 215–221, New York, NY, USA, July 2004. ACM. 22

[HLO$^+$10] Tommi Heikkinen, Tomas Lindén, Timo Ojala, Hannu Kukka, Marko Jurmu, and Simo Hosio. Lessons learned from the deployment and maintenance of ubi-hotspots. In *Proceedings of the 4th International Conference on Multimedia and Ubiquitous Engineering*, MUE '10, August 2010. 38

[HLSH09] Chris Harrison, Brian Y. Lim, Aubrey Shick, and Scott E. Hudson. Where to locate wearable displays?: Reaction time performance of visual alerts from tip to toe. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 941–944, New York, NY, USA, 2009. ACM. 21

[HM03] Elaine M. Huang and Elizabeth D. Mynatt. Semi-public displays for small, co-located groups. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 49–56, New York, NY, USA, 2003. ACM. 31, 90

[HOKK10] Akito Hyakutake, Koichiro Ozaki, Kris Makoto Kitani, and Hideki Koike. 3-d interaction with a large wall display using transparent markers. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, pages 97–100, New York, NY, USA, 2010. ACM. 48

[HPL$^+$13] Kiryong Ha, Padmanabhan Pillai, Grace Lewis, Soumya Simanta, Sarah Clinch, Nigel Davies, and Mahadev Satyanarayanan. The impact of mobile multimedia applications on data center consolidation. In *Proceedings of the IEEE International Conference on Cloud Engineering*, IC2E '13, 2013. i, 94

[HR08] Robert Hardy and Enrico Rukzio. Touch & interact: Touch-based interaction of mobile phones with displays. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '08, pages 245–254, New York, NY, USA, 2008. ACM. 43

[HRD11] John Hardy, Enrico Rukzio, and Nigel Davies. Real world responses to interactive gesture based public displays. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, MUM '11, pages 33–39, New York, NY, USA, December 2011. ACM. 65, 67, 70, 71, 73, 77, 78, 91, 308

[HRKS06] Paul Holleis, Enrico Rukzio, Thomas Kraus, and Albrecht Schmidt. Environment based messaging. In *Advances in Pervasive Computing 2006*. Austrian Computer Society, May 2006. 49

[HRS04] Elaine M. Huang, , Daniel M. Russell, and Alison E. Sue. Im here: Public instant messaging on large, shared displays for workgroup interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 279–286, New York, NY, USA, 2004. ACM. 32

[HRWP09] Robert Hardy, Enrico Rukzio, Matthias Wagner, and Massimo Paolucci. Exploring expressive nfc-based mobile phone interaction with large dynamic displays. In *Proceedings of the 2009 First International Workshop on Near Field Communication*, NFCs '09, pages 36–41, Washington, DC, USA, 2009. IEEE Computer Society. 43

[HS03] Lars Erik Holmquist and Tobias Skog. Informative art: Information visualization in everyday environments. In *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, GRAPHITE '03, pages 229–235, New York, NY, USA, 2003. ACM. 25, 26, 90

[HSS13] Roland Heuger, Sebastian Sadowski, and Florian Schulz. flourish - an interactive installation in public space. http://flourish.ahoi.in/ [Last accessed: April 2013], 2013. 45

[HW13] Steven Houben and Christian Weichel. Overcoming interaction blindness through curiosity objects. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '13, pages 1539–1544, New York, NY, USA, April 2013. ACM. 76

[IANAI13] The Internet Assigned Numbers Authority (IANA). Mime media types. http://www.iana.org/assignments/media-types [Last accessed: November 2013], 2013. 208

[IBR$^+$03] Shahram Izadi, Harry Brignull, Tom Rodden, Yvonne Rogers, and Mia Underwood. Dynamo: A public interactive surface supporting the cooperative sharing and exchange of media. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, pages 159–168, New York, NY, USA, 2003. ACM. 35, 84

[iMa13] iMatix Corporation. The intelligent transport layer - zeromq. http://www.zeromq.org/ [Last accessed: July 2013], 2007–2013. 195

[INF09] INFOSCREEN Austria, Gesellschaft fúr Stadtinformationsanlagen GmbH. Infoscreen - your city channel | infoscreen. http://www.infoscreen.at/www/homepage.php?lng=EN [Last accessed: January 2013], 2005–2009. 37

[INF12] INFOSCREEN NETWORKS PLC. Infoscreen networks. http://www.infoscreennetworks.com/ [Last accessed: February 2013], 2011–2012. 37

[Int] Intel Corporation. Intel® AIM Suite. https://aimsuite.intel.com/ [Last accessed: September 2013]. 307

[IRF01] Hiroshi Ishii, Sandia Ren, and Phil Frei. Pinwheels: Visualizing information flow in an architectural space. In *Extended Abstracts on*

Human Factors in Computing Systems, CHI '01, pages 111–112, New York, NY, USA, March 2001. ACM. 22

[IWB+98] Hiroshi Ishii, Craig Wisneski, Scott Brave, Andrew Dahley, Matt Gorbet, Brygg Ullmer, and Paul Yarin. ambientROOM: Integrating ambient media with architectural space. In *Conference Summary on Human Factors in Computing Systems*, CHI '98, pages 173–174, New York, NY, USA, April 1998. ACM. 22, 90

[JCD14] JCDecaux Group. Jcdecaux uk | no. 1 outdoor advertising company. `http://www.jcdecaux.co.uk/` [Last accessed: May 2014], 2014. 60, 61

[jF13] The jQuery Foundation. jQuery. `http://jquery.com/` [Last accessed: October 2013], 2013. 215

[JFHZ05] Nassim Jafarinaimi, Jodi Forlizzi, Amy Hurst, and John Zimmerman. Breakaway: An ambient display designed to change human behavior. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '05, pages 1945–1948, New York, NY, USA, April 2005. ACM. 23

[JMR+10] Giulio Jacucci, Ann Morrison, Gabriela T. Richard, Jari Kleimola, Peter Peltonen, Lorenza Parisi, and Toni Laitinen. Worlds of information: Designing for engagement at a public multi-touch display. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2267–2276, New York, NY, USA, 2010. ACM. 38, 72, 74

[JOIH08] Rui José, Nuno Otero, Shahram Izadi, and Richard Harper. Instant Places: Using bluetooth for situated interaction in public displays. *Pervasive Computing, IEEE*, 7, 2008. 42, 88, 90, 91, 183, 186, 191, 192

[JOMS06] Hao Jiang, Eyal Ofek, Neema Moraveji, and Yuanchun Shi. Direct Pointer: Direct manipulation for large-display interaction using hand-held cameras. In *Proceedings of the SIGCHI Conference on Human*

Factors in Computing Systems, CHI '06, pages 1107–1110, New York, NY, USA, April 2006. ACM. 46

[JPS$^+$12] Rui José, Hélder Pinto, Bruno Silva, Ana Melro, and Helena Rodrigues. Beyond interaction: Tools and practices for situated publication in display networks. In *Proceedings of the 2012 International Symposium on Pervasive Displays*, PerDis '12, New York, NY, USA, 2012. ACM. 88, 91, 238

[JPSM13] Rui José, Hélder Pinto, Bruno Silva, and Ana Melro. Instant Places: Using pins and posters as paradigms for content publication for situated displays. *Computer Graphics and Applications, IEEE*, 2013. 88, 238

[JVG$^+$01] Gavin Jancke, Gina Danielle Venolia, Jonathan Grudin, J. J. Cadiz, and Anoop Gupta. Linking public spaces: Technical and social issues. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 530–537, New York, NY, USA, March 2001. ACM. 19, 90

[Kal13] Martin Kaltenbrunner. TUIO. `http://www.tuio.org/` [Last accessed: May 2013], 2013. 47

[Kap01] Konstantin V. Kaplinsky. VNC tight encoder - data compression for VNC. In *Modern Techniques and Technology, 2001. MTT 2001. Proceedings of the 7th International Scientific and Practical Conference of Students, Post-graduates and Young Scientists*, pages 155–157, 2001. 81

[KBBC05] Martin Kaltenbrunner, Till Bovermann, Ross Bencina, and Enrico Costanza. Tuio - a protocol for table-top tangible user interfaces. In *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation*, GW 2005, 2005. 47

[KCDL12] Thomas Kubitza, Sarah Clinch, Nigel Davies, and Marc Langheinrich. Using mobile devices to personalize pervasive displays. In *Demo. at HotMobile '12*, 2012. i, 194, 238

[KCF$^+$06] Christian Kray, Keith Cheverst, Dan Fitton, Corina Sas, John Patterson, Mark Rouncefield, and Christoph Stahl. Sharing control of dispersed situated displays between nomadic and residential users. In *Proceedings of the 8th Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '06, pages 61–68, New York, NY, USA, 2006. ACM. 29

[KD03] Karrie Karahalios and Judith Donath. Telemurals: Catalytic connections for remote spaces. In Constantine Stephanidis and Julia A. Jacko, editors, *Human-Computer Interaction: Theory and Practice (Part 2)*, Human Factors and Ergonomics Series, pages 103–107. Erlbaum, 2003. 19

[KD04] Karrie Karahalios and Judith Donath. Telemurals: Linking remote spaces with social catalysts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 615–622, New York, NY, USA, April 2004. ACM. 19

[KGR08] Christian Kray, Areti Galani, and Michael Rohs. Facilitating opportunistic interaction with ambient displays. In *Workshop on Designing and Evaluating Mobile Phone-Based Interaction with Public Displays at CHI 2008*, 2008. 39

[KKK05] Christian Kray, Gerd Kortuem, and Antonio Krüger. Adaptive navigation support with public displays. In *Proceedings of the 10th international conference on Intelligent user interfaces*, IUI '05, pages 326–328, New York, NY, USA, 2005. ACM. 29

[KKK$^+$11] Hannu Kukka, Fabio Kruger, Vassilis Kostakos, Timo Ojala, and Marko Jurmu. Information to go: Exploring in-situ information pick-up "in the wild". In *Proceedings of the 13th IFIP TC 13 International Conference on Human-Computer Interaction - Volume Part II*, INTERACT '11, pages 487–504, Berlin, Heidelberg, 2011. Springer-Verlag. 52

[KKP$^+$04] Bill Kules, Hyunmo Kang, Catherine Plaisant, Anne Rose, and Ben Shneiderman. Immediate usability: a case study of public access design for a community photo library. *Interacting with Computers*, 16(6):1171–1193, 2004. 72, 74

[KNF09] Hideki Koike, Wataru Nishikawa, and Kentaro Fukuchi. Transparent 2-D markers on an LCD tabletop system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 163–172, New York, NY, USA, April 2009. ACM. 48

[KOK$^+$13] Hannu Kukka, Heidi Oja, Vassilis Kostakos, Jorge Gonçalves, and Timo Ojala. What makes you click: Exploring visual signals to entice interaction on public displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1699–1708, New York, NY, USA, April 2013. ACM. 68, 72, 91

[Kov13] Anton Kovalyov. JSHint. `http://www.jshint.com/` [Last accessed: October 2013], 2013. 216

[KS02] Michael A. Kozuch and Mahadev Satyanarayanan. Internet suspend/resume. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, pages 40–46, 2002. 228, 239, 291

[KSFS05] Tim Kindberg, Mirjana Spasojevic, Rowanne Fleck, and Abigail Sellen. The ubiquitous camera: An in-depth study of camera phone use. *Pervasive Computing, IEEE*, 4(2):42–50, April 2005. 50

[KW06] Satoshi Kuribayashi and Akira Wakita. PlantDisplay: Turning houseplants into ambient display. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology*, ACE '06, New York, NY, USA, 2006. ACM. 22

[Lan02] Marc Langheinrich. A privacy awareness system for ubiquitous computing environments. In *Proceedings of the 4th International Conference on Ubiquitous Computing*, Ubicomp '02, pages 237–245. Springer-Verlag, 2002. 192

[LBF99] P. Ljungstrand, S. Bjork, and J. Falk. The wearboy: a platform for low-cost public wearable devices. In *Proceedings of the Third International Symposium on Wearable Computers.*, pages 195–196, October 1999. 20, 21

[Lei13] Jannis Leidel. django-discover-runner. `https://github.com/jezdez/django-discover-runner` [Last accessed: October 2013], 2012-2013. 216

[LKS$^+$11] Jae Yeol Lee, Min Seok Kim, Dong Woo Seo, Chil-Woo Lee, Jae Sung Kim, and Sang Min Lee. Dual interactions between multi-display and smartphone for collaborative design and sharing. In *Proceedings of IEEE Virtual Reality (VR) Conference 2011*, pages 221–222, 2011. 56, 90

[LKS$^+$12] Jae Yeol Lee, Min Seok Kim, Dong Woo Seo, Chil-Woo Lee, Jae Sung Kim, and Sang Min Lee. Smart and space-aware interactions using smartphones in a shared space. In *Proceedings of the 14th international conference on Human- computer interaction with mobile devices and services companion*, MobileHCI '12, pages 53–58, New York, NY, USA, 2012. ACM. 56

[LMEA11] Marc Langheinrich, Nemanja Memarovic, Ivan Elhart, and Florian Alt. Autopoiesic content: A conceptual model for enabling situated self-generative content for public displays. In *First International Workshop on Pervasive Urban Applications (PURBA) at Pervasive 2011*, 2011. 37, 78

[LSI$^+$06] Jaana Leikas, Hanna Stromberg, Veikko Ikonen, Riku Suomela, and Juhani Heinila. Multi-user mobile applications and a public display: Novel ways for social interaction. In *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications*, PerCom '06, March 2006. 54

[Mac13] Matt Mackall. Mercurial scm. `http://mercurial.selenic.com/` [Last accessed: October 2013], 2000–2013. 196, 216

[MAMS10] Jörg Müller, Florian Alt, Daniel Michelis, and Albrecht Schmidt. Requirements and design space for interactive public displays. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 1285–1294, New York, NY, USA, 2010. ACM. 75

[McC02] Joseph F. McCarthy. Using public displays to create conversation opportunities. In *Public, Community and Situated Displays (Workshop at CSCW 2002)*, New Orleans, LA, USA, 2002. 34, 83

[McC03] Joseph F. McCarthy. Promoting a sense of community with ubiquitous peripheral displays. In Kenton O'Hara, Mark Perry, Elizabeth Churchill, and Daniel M. Russel, editors, *Public and Situated Displays – Social and Interactional Aspects of Shared Display Technologies*, pages 283–308. Kluwer, 2003. 34, 83

[MCH08] Joseph F. McCarthy, Ben Congleton, and F. Maxwell Harper. The context, content & community collage: Sharing personal digital media in the physical workplace. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, CSCW '08, pages 97–106, New York, NY, USA, 2008. ACM. 33, 85, 96, 186

[MCL01] Joseph F. McCarthy, Tony J. Costa, and Edy S. Liongosari. Unicast, outcast & groupcast: Three steps toward ubiquitous, peripheral displays. In *Proceedings of the 3rd International Conference on Ubiquitous Computing*, Ubicomp '01, pages 332–345, London, UK, 2001. Springer-Verlag. 28, 34, 83, 96, 97

[MEBK09] Jörg Müller, Juliane Exeler, Markus Buzeck, and Antonio Krüger. Reflectivesigns: Digital signs that adapt to audience attention. In *Proceedings of the 7th International Conference on Pervasive Computing*, Pervasive '09, pages 17–24, Berlin, Heidelberg, May 2009. Springer-Verlag. 39

[MEL11] Nemanja Memarovic, Ivan Elhart, and Marc Langheinrich. FunSquare: First experiences with autopoiesic content. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, MUM

’11, pages 175–184, New York, NY, USA, December 2011. ACM. 37, 78

[MEM+13] Nemanja Memarovic, Ivan Elhart, Andrea Michelotti, Elisa Rubegni, and Marc Langheinrich. Social networked displays: Integrating networked public displays with social media. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp ’13 Adjunct, pages 55–58, New York, NY, USA, 2013. ACM. 238

[Mey13] Carl Meyer. django-model-utils – django model mixins and utilities. https://django-model-utils.readthedocs.org [Last accessed: October 2013], 2013. 213

[MHT04] Kento Miyaoku, Suguru Higashino, and Yoshinobu Tonomura. C-Blink: A hue-difference-based light signal marker for large screen interaction via any mobile terminal. In *Proceedings of the 17th Annual ACM symposium on User Interface Software and Technology*, UIST ’04, pages 147–156, New York, NY, USA, 2004. ACM. 47

[Mic12] Microsoft. Kinect - Xbox.com. http://www.xbox.com/en-US/kinect/ [Last accessed: December 2012], 2012. 56

[Mic14] Microsoft. [MS-RDPBCGR]: Remote Desktop Protocol: Basic connectivity and graphics remoting. http://msdn.microsoft.com/en-us/library/cc240445.aspx [Last accessed: May 2014], 2007–2014. 80, 191

[Mik13] Mateusz Mikusz. Building and using an app store to support public display users. Diploma thesis, University of Stuttgart, 2013. 216

[MJP08] Ann Morrison, Giulio Jacucci, and Peter Peltonen. Citywall: Limitations of a multi-touch environment. In *Proceedings of the 2008 Workshop on Designing Multi-Touch Interaction Techniques for Coupled Public and Private Displays (at AVI 2008).*, PPD ’08, 2008. 38

[MK06] Jörg Müller and Antonio Krüger. Towards situated public displays as multicast systems. In *UbiqUM 2006 Workshop on Ubiquitous User Modeling at the 17th European Conference on Artificial Intelligence*, 2006. 59

[MK07a] Jörg Müller and Antonio Krüger. How much to bid in digital signage advertising auctions. In *Adjunct proceedings of Pervasive*, 2007. 59, 171

[MK07b] Jörg Müller and Antonio Krüger. User profiling for generating bids in digital signage advertising auctions. In *Adjunct Proceedings of User Modeling*. Springer-Verlag, 2007. 59

[MK09] Jörg Müller and Antonio Krüger. MobiDiC: Context adaptive digital signage with coupons. In *Proceeedings of the European Conference on Ambient Intelligence*, AmI '09, pages 24–33, Berlin, Heidelberg, 2009. Springer-Verlag. 59, 90

[MKHS08] Hema Mahato, Dagmar Kern, Paul Holleis, and Albrecht Schmidt. Implicit personalization of public environments using Bluetooth. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '08, pages 3093–3098, New York, NY, USA, 2008. ACM. 42, 87, 140, 192

[MKK07] Jörg Müller, Antonio Krüger, and Tsvi Kuflik. Maximizing the utility of situated public displays. In *Adjunct Proceedings of User Modeling*, pages 395–399. Springer-Verlag, 2007. 59

[MLA11] Nemanja Memarovic, Marc Langheinrich, and Florian Alt. Connecting people through content – promoting community identity cognition through people and places. In *Proceedings of Community Informatics*, 2011. 37, 78, 79

[MLA12a] Nemanja Memarovic, Marc Langheinrich, and Florian Alt. Interacting places – a framework for promoting community interaction and place awareness through public displays. In *Adjunct Proceedings of the Tenth*

Annual IEEE International Conference on Pervasive Computing and Communications (PerCom). IEEE Computer Society, March 2012. 37

[MLA12b] Nemanja Memarovic, Marc Langheinrich, and Florian Alt. The interacting places framework: Conceptualizing public display applications that promote community interaction and place awareness. In *Proceedings of the 2012 International Symposium on Pervasive Displays*, PerDis '12, New York, NY, USA, 2012. ACM. 37

[MLR$^+$12] Nemanja Memarovic, Marc Langheinrich, Elisa Rubegni, Andreia David, and Ivan Elhart. Designing "interacting places" for a student community using a communicative ecology approach. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, MUM '12, New York, NY, USA, December 2012. ACM. 37

[MM11] Daniel Michelis and Jörg Müller. The audience funnel: Observations of gesture based interaction with multiple large displays in a city center. *International Journal of Human-Computer Interaction*, 27(6):562–579, 2011. 63, 64, 68, 71, 73, 78, 91

[MMS$^+$04] Joseph F. McCarthy, David W. McDonald, Suzanne Soroczak, David H. Nguyen, and Al M. Rashid. Augmenting the social space of an academic conference. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 39–48. ACM, New York, NY, USA, 2004. 36

[Mob13] Mobile Radicals, Lancaster University. Mobile radicals. http://www.mobileradicals.com/projects [Last accessed: April 2013], 2013. 49

[MPK07] Jörg Müller, Oliver Paczkowski, and Antonio Krüger. Situated public news and reminder displays. In *Proceedings of the European Conference on Ambient Intelligence*, AmI '07, pages 248–265, Berlin, Heidelberg, 2007. Springer-Verlag. 39

[MRS06] Keith Mitchell, Nicholas J. P. Race, and Michael Suggitt. iCapture: Facilitating spontaneous user-interaction with pervasive displays using smart devices. In *Proceedings of Workshop on Pervasive Mobile Interaction Devices - Mobile Devices as Pervasive User Interfaces and Interaction Devices*, PERMID '06, 2006. 50, 90

[MS97] Jennifer Mankoff and Bill N. Schilit. Supporting knowledge workers beyond the desktop with palplates. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, pages 550–551. ACM, March 1997. 28

[MSK07] Jörg Müller, Alex Schlottmann, and Antonio Krüger. Self-optimising digital signage advertising. In *Demo at Ubicomp 2007*, 2007. 59

[MWB$^+$12] Jörg Müller, Robert Walter, Gilles Bailly, Michael Nischt, and Florian Alt. Looking glass: A field study on noticing interactivity of a shop window. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 297–306, New York, NY, USA, May 2012. ACM. 68, 73, 75, 78, 91

[MWE$^+$09] Jörg Müller, Dennis Wilmsmann, Juliane Exeler, Markus Buzeck, Albrecht Schmidt, Tim Jay, and Antonio Krüger. Display blindness: The effect of expectations on attention towards digital signage. In *Proceedings of the 7th International Conference on Pervasive Computing*, Pervasive '09, pages 1–8, Berlin, Heidelberg, May 2009. Springer-Verlag. 1, 3, 39, 66, 67, 79, 91, 267, 309

[MWP10] Jörg Müller, Lena Wickenkamp, and Denise Paradowski. Why do shop owners install digital signage? In *Proceedings of the 3rd Workshop on Pervasive Advertising and Shopping*, 2010. 56, 104

[Nin13] Nintendo. Wii official site at nintendo. `http://www.nintendo.com/wii` [Last accessed: January 2013], 2013. 49

[NMS10] Anders Nickelsen, Miquel Martin, and Hans-Peter Schwefel. Service migration protocol for NFC links. In *Proceedings of the 16th EUNICE/*

IFIP WG 6.6 Conference on Networked Services and Applications: Engineering, Control and Management, EUNICE'10, pages 41–50, Berlin, Heidelberg, 2010. Springer-Verlag. 44, 54

[NoM14] NoMachine S.á r.l. NoMachine - free remote desktop for everybody. `https://www.nomachine.com/` [Last accessed: April 2014], 2002-2014. 80

[NTDM00] D.H. Nguyen, J. Tullio, T. Drewes, and E.D. Mynatt. Dynamic door displays. GVU Technical Report GIT-GVU-00-30, Georgia Institute of Technology, GVU, 2000. 28

[OBM+13] Ronald Oussoren, Bill Bumgarner, Steve Majewski, Lele Gaifax, et al. PyObjC – the Python $\leftrightarrow$ Objective-C bridge. `http://pythonhosted.org/pyobjc/` [Last accessed: October 2013], 2013. 196

[OHU+05] Kenton O'Hara, Richard Harper, Axel Unger, James Wilkes, Bill Sharpe, and Marcel Jansen. TxtBoard: From text-to-person to text-to-home. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '05, pages 1705–1708, New York, NY, USA, April 2005. ACM. 40

[Oka04] Daisuke Okabe. Emergent social practices, situations and relations through everyday camera phone use. In *Proceedings of the 2004 International Conference on Mobile Communication and Social Change*, pages 1–19, October 2004. 50

[OKK+12] Timo Ojala, Vassilis Kostakos, Hannu Kukka, Tommi Heikkinen, Tomas Lindén, Marko Jurmu, Simo Hosio, Fabio Kruger, and Daniele Zanni. Multipurpose interactive public displays in the wild: Three years later. *Computer, IEEE*, 45(5):42–49, May 2012. 38, 71, 72, 73, 91, 92, 99, 182

[OKL+10] Timo Ojala, Hannu Kukka, Tomas Lindén, Tommi Heikkinen, Marko Jurmu, Simo Hosio, and Fabio Kruger. Ubi-hotspot 1.0: Large-scale long-term deployment of interactive public displays in a city center. In *Proceedings of the 2010 Fifth International Conference on Internet and Web Applications and Services*, ICIW '10, pages 285–294, Washington, DC, USA, 2010. IEEE Computer Society. 38, 90

[OLJ+04] Kenton O'Hara, Matthew Lipson, Marcel Jansen, Axel Unger, Huw Jeffries, and Pater Macer. Jukola: Democratic music choice in a public space. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, DIS '04, pages 145–154, New York, NY, USA, July 2004. ACM. 35

[OPL03] Kenton O'Hara, Mark Perry, and Simon Lewis. Situated Web signs and the ordering of social action. In Kenton O'Hara, Mark Perry, Elizabeth Churchill, and Daniel M. Russel, editors, *Public and Situated Displays – Social and Interactional Aspects of Shared Display Technologies*, pages 105–140. Kluwer, 2003. 29

[Ora] Oracle Corporation. Mysql:: The world's most popular open source database. http://www.mysql.com/ [Last accessed: October 2013]. 212

[oT13a] Queensland University of Technology. Welcome to Nnub. http://nnub.net/ [Last accessed: January 2013], 2013. 39

[OT13b] Mark Otto and Jacob Thornton. Bootstrap. http://getbootstrap.com/2.3.2/ [Last accessed: October 2013], 2011–2013. 215

[Pac13] Pacific Software Publishing, Inc. ClockLink.com - free flash world clock & free HTML5 clocks. http://www.clocklink.com/gallery.php?category=HTML5 [Last accessed: November 2013], 2013. 228

[PaNR13] Manuel Pereira, Maria Jo ao Nicolau, and Helena Rodrigues. Application synchronisation on public displays based on PubSubHubbub. In *Proceedings of Simpósio de Informática*, INFORUM, September 2013. 182

[PBW05] Trevor Pering, Rafael Ballagas, and Roy Want. Spontaneous marriages of mobile devices and interactive spaces. *Communications of the ACM*, 48(9):53–59, 2005. 44, 53, 89

[PD-13] PD-NET. PD-NET. http://pd-net.org/ [Last accessed: November 2013], 2013. 237

[PDJS06] Terry Payne, Ester David, Nicholas R. Jennings, and Matthew Sharifi. Auction mechanisms for efficient advertisement selection on public displays. In *Proceedings of the 17th European Conference on Artificial Intelligence*, ECAI 2006, pages 285–289, Amsterdam, The Netherlands, August–September 2006. IOS Press. 59

[Pen13] Raymond Penners. django-allauth. https://github.com/pennersr/django-allauth [Last accessed: October 2013], 2013. 215

[PG13a] The PHP Group. Html_template_it. http://pear.php.net/package/HTML_Template_IT [Last accessed: November 2013], 2001–2013. 228

[PG13b] The PHP Group. PHP: DOM – manual. http://www.php.net/manual/en/book.dom.php [Last accessed: October 2013], 2001–2013. 205

[PG13c] The PHP Group. PHP: Hypertext Preprocessor. http://www.php.net/ [Last accessed: October 2013], 2001–2013. 205, 226, 228

[Pik91] Rob Pike. $8\frac{1}{2}$, the Plan 9 window system. In *Proceedings of the Summer 1991 USENIX Conference*, pages 257–265, June 1991. 80

[PJO09] Nick Pears, Daniel G. Jackson, and Patrick Olivier. Smart phone interaction with registered displays. *Pervasive Computing, IEEE*, 8(2):14–21, April 2009. 46, 47

[PKS$^+$08] Peter Peltonen, Esko Kurvinen, Antti Salovaara, Giulio Jacucci, Tommi Ilmonen, John Evans, Antti Oulasvirta, and Petri Saarikko.

It's mine, don't touch!: Interactions at a large multi-touch display in a city centre. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1285–1294, New York, NY, USA, 2008. ACM. 38, 69, 70, 75, 78, 91

[Pla13] Dieter Plaetinck. Uzbl – web interface tools which adhere to the unix philosophy. `http://www.uzbl.org/` [Last accessed: November 2013], 2013. 205

[POJ08] Nick Pears, Patrick Olivier, and Dan Jackson. Display registration for device interaction - a proof of principle prototype. In *Proceedings of the 3rd International Conference on Computer Vision Theory and Applications*, VISAPP '08, pages 446–451. The Institute for Systems and Technologies of Information, Control and Communication (INSTICC), January 2008. 46

[PPTT90] Rob Pike, Dave Presotto, Ken Thompson, and Howard Trickey. Plan 9 from Bell Labs. In *Proceedings of the Summer 1990 UKUUG Conference*, pages 1–9, July 1990. 80

[PRS+03] Thorsten Prante, Carsten Röcker, Norbert Streitz, Richard Stenzel, Carsten Magerkurth, Daniel van Alphen, and Daniela Plewe. Hello.Wall – beyond ambient displays. In *Proceedings of the 5th International Conference on Ubiquitous Computing*, Ubicomp '03, pages 277–278, Seattle, WA, USA, October 2003. Springer. 23, 62, 63

[PSJ+07] Peter Peltonen, Antti Salovaara, Giulio Jacucci, Tommi Ilmonen, Carmelo Ardito, Petri Saarikko, and Vikram Batra. Extending large-scale event participation with user-created mobile media on a public display. In *Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia*, MUM '07, pages 131–138, New York, NY, USA, 2007. ACM. 38

[Pyt13a] Python Software Foundation. flake8 2.0. `https://pypi.python.org/pypi/flake8` [Last accessed: October 2012], 1990–2013. 216

[Pyt13b] Python Software Foundation. PEP 8 - style guide for Python code. http://www.python.org/dev/peps/pep-0008/ [Last accessed: October 2013], 1990–2013. 196

[Pyt13c] Python Software Foundation. pep8 - Python style guide checker. https://pypi.python.org/pypi/pyflakes [Last accessed: October 2013], 1990–2013. 196, 216

[Pyt13d] Python Software Foundation. pyflakes 0.7.3 – passive checker of Python programs. https://pypi.python.org/pypi/pyflakes [Last accessed: October 2013], 1990–2013. 196, 216

[Pyt13e] Python Software Foundation. Python programming language. http://python.org/ [Last accessed: July 2013], 1990–2013. 115, 195, 205, 228

[Pyt13f] Python Software Foundation. unittest – unit testing framework. http://docs.python.org/3/library/unittest.html [Last accessed: October 2013], 1990–2013. 196

[Pyt13g] Python Software Foundation. xml.etree.ElementTree – the ElementTree XML API. http://docs.python.org/2/library/xml.etree.elementtree.html [Last accessed: November 2013], 1990–2013. 205

[Pyt13h] Python Software Foundation. doctest Ñ test interactive Python examples. http://docs.python.org/3/library/doctest.html [Last accessed: October 2013], 1990-2013. 196

[R&03] France Telecom R&D. Wearable communications : Optical fibres. http://www.studio-creatif.com/Gb/Vet/Vet02Prototypes05Fr.htm [Last accessed: November 2012], 2003. 21

[RB08] Fiona Redhead and Margot Brereton. Getting to the nub of neighbourhood interaction. In *Proceedings of the Tenth Anniversary Conference* *on Participatory Design 2008*, PDC '08, pages 270–273, Indianapolis, IN, USA, 2008. Indiana University. 39

[RB09] Fiona Redhead and Margot Brereton. Designing interaction for local communications: An urban screen study. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II*, INTERACT '09, pages 457–460, Berlin, Heidelberg, 2009. Springer-Verlag. 39

[RBMH94] Tristan Richardson, Frazer Bennett, Glenford Mapp, and Andy Hopper. Teleporting in an X Window system environment. *Personal Communications, IEEE*, 1(3):6–13, 1994. 82

[RC02] Anand Ranganathan and Roy H. Campbell. Advertising in a pervasive computing environment. In *Proceedings of the 2nd International Workshop on Mobile Commerce*, WMC '02, pages 10–14, New York, NY, USA, 2002. ACM. 57

[RDO$^+$05] Ismo K. Rakkolainen, Stephen DiVerdi, Alex Olwal, Nicola Candussi, Tobias Hüllerer, Markku Laitinen, Mika Piirto, and Karri Palovuori. The interactive FogScreen. In *ACM SIGGRAPH 2005 Emerging Technologies*, SIGGRAPH '05, New York, NY, USA, 2005. ACM. 23

[RDS02] Daniel M. Russell, Clemens Drews, and Alison Sue. Social aspects of using large public interactive displays for collaboration. In *Proceedings of the 4th International Conference on Ubiquitous Computing*, Ubicomp '02, pages 229–236, London, UK, 2002. Springer-Verlag. 34, 74, 78

[Red09] Red Hat, Inc. Spice remote computing protocol definition v1.0. `http://www.spice-space.org/docs/spice_protocol.pdf` [Last accessed: May 2014], 2009. 80

[REE$^+$06] Ismo K. Rakkolainen, Tanju Erdem, Çiğdem Erdem, Mehmet Özkan, and Markku Laitinen. Interactive "immaterial" screen for performing arts. In *Proceedings of the 14th Annual ACM International Conference* *on Multimedia*, MULTIMEDIA '06, pages 185–188, New York, NY, USA, 2006. ACM. 23

[Rem12] Remote Media Limited. signagelive — cloud-based digital signage software. http://www.signagelive.com/ [Last accessed: September 2012], 2012. 3, 156

[RG01] Daniel M. Russell and Rich Gossweiler. On the design of personal & communal large information scale appliances. In *Proceedings of the 3rd International Conference on Ubiquitous Computing*, Ubicomp '01, pages 354–361, London, UK, 2001. Springer-Verlag. 34

[RG04] Michael Rohs and Beat Gfeller. Using camera-equipped mobile phones for interacting with real-world objects. In *Advances in Pervasive Computing*, pages 265–271, 2004. 45

[RHM$^+$10] Yvonne Rogers, William R. Hazlewood, Paul Marshall, Nick Dalton, and Susanna Hertrich. Ambient influence: Can twinkly lights lure and abstract representations trigger behavioral change? In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, Ubicomp '10, pages 261–270, New York, NY, USA, 2010. ACM. 23, 40

[RL07] Ismo K. Rakkolainen, , and Artur K. Lugmayr. Immaterial display for interactive advertisements. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology*, ACE '07, pages 95–98, New York, NY, USA, 2007. ACM. 23, 57

[RLHS04] Josephine Reid, Mathew Lipson, Jenny Hyams, and Kate Shaw. "fancy a schmink?": a novel networked game in a café. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology*, ACE '04, pages 18–23, New York, NY, USA, June 2004. ACM. 35

[RMH09] Enrico Rukzio, Michael Mueller, and Robert Hardy. Design, implementation and evaluation of a novel public display for pedestrian navigation: The Rotating Compass. In *Proceedings of the 27th International*

Conference on Human factors in computing systems, CHI '09, pages 113–122, New York, NY, USA, 2009. ACM. 50

[RMKA08] Gustavo Ramírez-González, Mario Muñoz-Organero, Carlos Delgado Kloos, and Ángela Chantre Astaiza. Exploring NFC interactive panel. In *Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, Mobiquitous '08, pages 32:1–32:2, Brussels, Belgium, 2008. ICST. 43, 44

[RML11] Elisa Rubegni, Nemanja Memarovic, and Marc Langheinrich. Talking to strangers: Using large public displays to facilitate social interaction. In *Proceedings of the 14th International Conference on Human-Computer Interaction*, HCI '11, pages 195–204. Springer-Verlag, July 2011. 36

[Rod99] Roy Rodenstein. Employing the periphery: the window as interface. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '99, pages 204–205, New York, NY, USA, May 1999. ACM. 23

[Row14] Dan Rowinski. The reviews are in: Android apps outshone iOS apps in 2013. `http://readwrite.com/2014/01/31/android-ios-app-quality-utest-applause` [Last accessed: May 2014], January 2014. 288

[RP02] Ismo K. Rakkolainen and Karri Palovuori. WAVE – a walk-thru virtual environment. In *CD Proceedings of the 6th Immersive Projection Technology Symposium*, In association with IEEE VR 2002, March 2002. 23

[RP04] Ismo K. Rakkolainen and Karri Palovuori. Interactive digital FogScreen. In *Proceedings of the 3rd Nordic Conference on Human-Computer Interaction*, NordiCHI '04, pages 459–460, New York, NY, USA, 2004. ACM. 23

[RP05] Ismo K. Rakkolainen and Karri Palovuori. Laser scanning for the interactive walk-through FogScreen. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '05, pages 224–226, New York, NY, USA, 2005. ACM. 23

[RSH00] Johan Redström, Tobias Skog, and Lars Hallnäs. Informative art: Using amplified artworks as information displays. In *Proceedings of Designing Augmented Reality Environments*, DARE '00, pages 103–114, New York, NY, USA, 2000. ACM. 24, 25, 90

[RSK05] Enrico Rukzio, Albrecht Schmidt, and Antonio Krüger. The rotating compass: A novel interaction technique for mobile navigation. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '05, pages 1761–1764, New York, NY, USA, April 2005. ACM. 50

[RSWH98] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper. Virtual network computing. *Internet Computing, IEEE*, 2(1):33–38, January–February 1998. 80, 82

[RTD04] Daniel M. Russell, Jay P. Trimble, and Andreas Dieberger. The use patterns of large, interactive display surfaces: Case studies of media design and use for blueboard and merboard. In *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) – Track 4 – Volume 4*, HICSS '04, Washington, DC, USA, 2004. IEEE Computer Society. 34, 74

[RTW02] Daniel M. Russell, Jay P. Trimble, and Roxana Wales. Two paths from the same place: Task driven and human-centred evolution of a group information surface. In *Proceedings of the Make IT Easy Conference*, 2002. 34

[SA97] Bill Segall and David Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings of AUUG97*, 1997. 11

[SBA07] Daniel Stødle, John Markus Bjørndalen, and Otto J. Anshus. Decentralizing the VNC model for improved performance on wall-sized, high-resolution tiled displays. In *Proceedings of the Norwegian Informatics Conference*, NIK 2007, pages 53–64, 2007. 81

[SBCD09] Mahadev Satyanarayanan, Paramvir Bahl, Ramon Caceres, and Nigel Davies. The case for VM-based cloudlets in mobile computing. *Internet Computing, IEEE*, 8(4), 2009. 53, 89

[Sca14] Scala. Scala — digital signage software. `http://www.scala.com/` [Last accessed: April 2014], 2014. 3, 60

[SCFH12] J. She, J. Crowcroft, H. Fu, and P-H. Ho. Smart signage: A draggable cyber-physical broadcast/multicast media system. In *Proceedings of the IEEE International Conference on Cyber, Physical and Social Computing*, CPSCom 2012, 2012. 52

[SFD06a] Oliver Storz, Adrian Friday, and Nigel Davies. Supporting content scheduling on situated public displays. *Computers & Graphics*, 30(5):681–691, 2006. 8, 39, 90, 161, 166, 178

[SFD$^+$06b] Oliver Storz, Adrian Friday, Nigel Davies, Joe Finney, Corina Sas, and Jennifer Sheridan. Public ubiquitous computing systems: Lessons from the e- Campus display deployments. *Pervasive Computing, IEEE*, 3(5):40–47, 2006. 8, 26, 39, 99, 140, 161

[SG86] Robert W. Scheifler and Jim Gettys. The X window system. *ACM Transactions on Graphics*, 5(2):79–109, April 1986. 80, 82

[SG02] Dave Snowdon and Antonietta Grasso. Diffusing information in organizational settings: Learning from experience. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 331–338, New York, NY, USA, 2002. ACM. 30

[SH00] Tobias Skog and Lars Erik Holmquist. WebAware: Continuous visualization of web site activity in a public space. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '00, pages 351–352, New York, NY, USA, 2000. ACM. 24

[SHZF10] Andreas Sippl, Clemens Holzmann, Doris Zachhuber, and Alois Ferscha. Real-time gaze tracking for public displays. In *Proceedings of the First International Joint Conference on Ambient Intelligence*, AmI '10, pages 167–176, Berlin, Heidelberg, 2010. Springer-Verlag. 308

[SI01] Garth B. D. Shoemaker and Kori M. Inkpen. Single display privacyware: Augmenting public displays with private information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 522–529, New York, NY, USA, March 2001. ACM. 55

[SJE11] Rich Snow, Matt Jones, and Parisa Eslambolchilar. Projecting wonderment: Magic through ar. `http://cs.swan.ac.uk/~csrs/publications/ProjectingWonderment.pdf` [Last accessed: April 2013], 2011. 27, 101

[SK12] Henrik Sørensen and Jesper Kjeldskov. Immediate user interface adaption in multi-device environments. In *Proceedings of the 2012 Workshop on Infrastructure and Design Challenges of Couple Display Visual Interfaces at the International Working Conference on Advanced Visual Interfaces (AVI 2012)*, PPD '12, pages 14–17, May 2012. 54

[SKM09a] Martin Strohbach, Ernö Kovacs, and Miquel Martin. Pervasive display networks – real-world internet services for public displays. In *Adjunct Proceedings of the 4th European Conference on Smart Sensing and Context (EuroSSC)*, pages 39–42, 2009. 2, 3, 60

[SKM09b] Martin Strohbach, Ernö Kovacs, and Miquel Martin. Towards pervasively adapting display networks. In *Proceedings of the 1st International Workshop on Pervasive Advertising (held in conjunction with Pervasive 2009)*, May 2009. 2, 3, 51, 60, 91, 192

[Sko04] Tobias Skog. Activity wallpaper: Ambient visualization of activity information. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, DIS '04, pages 325–328, New York, NY, USA, July 2004. ACM. 26

[SLO$^+$06] Luis Sanchez, Jorge Lanza, Rasmus Olsen, Martin Bauer, and Marc Girod-Genet. A generic context management framework for personal networking environments. In *Proceedings of 3rd Annual International Conference on Mobile and Ubiquitous Systems*, pages 1–8, July 2006. 60

[SMA13] SMART Technologies. Smart board interactive whiteboards. http://www.smarttech.com/SmartBoard [Last accessed: January 2013], 2013. 30, 31

[SO05] Jürgen Scheible and Timo Ojala. MobiLenin – combining a multi-track music video, personal mobile phones and a public display into multi-user interactive entertainment. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, pages 199–208, May 2005. 45

[SOC08] Jürgen Scheible, Timo Ojala, and Paul Coulton. Mobitoss: a novel gesture based interface for creating and sharing mobile multimedia art on large public displays. In *Proceedings of the 16th ACM international conference on Multimedia*, MM '08, pages 957–960, New York, NY, USA, 2008. ACM. 49, 104

[Son14] Sony Electronics Inc. Ziris^{TM} digital signage software. http://pro.sony.com/bbsc/ssr/cat-digitalsignage/resource.solutions.bbsccms-assets-cat-digsignagedev-solutions-Ziris.shtml [Last accessed: April 2014], 2005–2014. 3, 61, 152, 154, 161

[SRP$^+$03] Norbert A. Streitz, Carsten Röcker, Thorsten Prante, Richard Stenzel, and Daniel van Alphen. Situated interaction with ambient information: Facilitating awareness and communication in ubiquitous work environments. In *Proceedings of the Tenth International Conference on Human-Computer Interaction*, HCI International 2003, June 2003. 23, 62, 63

[SSB06] Richard Sharp, James Scott, and Alastair R. Beresford. Secure mobile computing via public terminals. In *Proceedings of the 4th International*

Conference on Pervasive Computing, Pervasive '06, pages 238–253, Berlin, Heidelberg, 2006. Springer-Verlag. 55, 191

[Sta13] Thad Starner. Project Glass: Reducing the time between intention and action. *Pervasive Computing, IEEE*, 11(2), 2013. 234

[Sto08] Oliver Storz. *A Distributed Systems Infrastructure for Open Public Display Research Networks*. PhD thesis, Lancaster University, 2008. 8, 39

[Str13] Ströer Digital Media GmbH. Infoscreen. http://www.stroeerdigital.de/en/infoscreen/ [Last accessed: January 2013], 2013. 37, 90

[Sun09] Sun Microsystems, Inc. VirtualBox. http://www.virtualbox.org/ [Last accessed: November 2009], 2009. 116

[SWS01] Nitin Sawhney, Sean Wheeler, and Chris Schmandt. Aware community portals: Shared information appliances for transitional spaces. *Personal Ubiquitous Computing*, 5(1):66–70, January 2001. 30

[SWS09] Alireza Sahami Shirazi, Christian Winkler, and Albrecht Schmidt. Flashlight interaction: a study on mobile phone interaction techniques with large displays. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '09, New York, NY, USA, 2009. ACM. 47

[TC09] Nick Taylor and Keith Cheverst. Social interaction around a rural community photo display. *International Journal of Human-Computer Studies*, 67(12), December 2009. 39, 99, 140

[TC10] Nick Taylor and Keith Cheverst. Creating a rural community display with local engagement. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, DIS '10, pages 218–227, New York, NY, USA, 2010. ACM. 39

[TC12] Nick Taylor and Keith Cheverst. Supporting community awareness with interactive displays. *Computer, IEEE*, 45(5):26–32, 2012. 39

[TCF+07] Nick Taylor, Keith Cheverst, Dan Fitton, Nicholas J. P. Race, Mark Rouncefield, and Connor Graham. Probing communities: Study of a village photo display. In *Proceedings of the 2007 Australasian Computer-Human Interaction Conference*, OzCHI '07. ACM, November 2007. 39

[Ter14] Teradici Corporation. Teradici PCoIP solutions. http://www.teradici.com/ [Last accessed: May 2014], 2014. 80, 81

[TFB+08] Anthony Tang, Mattias Finke, Michael Blackstock, Rock Leung, Meghan Deutscher, and Rodger Lea. Designing for bystanders: Reflections on building a public digital forum. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 879–882, New York, NY, USA, 2008. ACM. 41, 65, 70, 71, 77

[The05] ThePooch. metamorphosis. http://web.archive.org/web/20060709215853/http://www.thepooch.com/Events/metamorphosis.htm [Last accessed: May 2013], 2005. 26

[The14] The Apache Software Foundation. ab - Apache HTTP server benchmarking tool. http://httpd.apache.org/docs/current/programs/ab.html [Last accessed: May 2014], 2014. 288

[URB13] URBANSCREEN. Urbanscreen | site-specific projections. http://www.urbanscreen.com/ [Last accessed: May 2013], 2010–2013. 27

[VB04] Daniel Vogel and Ravin Balakrishnan. Interactive public ambient displays: Transitioning from implicit to explicit, public to personal, interaction with multiple users. In *Proceedings of the 17th Annual ACM symposium on User Interface Software and Technology*, UIST '04, pages 137–146, New York, NY, USA, 2004. ACM. 26, 27, 63

[VBG13] Mélodie Vidal, Andreas Bulling, and Hans Gellersen. Pursuits: Spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 439–448, New York, NY, USA, 2013. ACM. 308

[VCBE08] Tamas Vajk, Paul Coulton, Will Bamford, and Reuben Edwards. Using a mobile phone as a "wii-like" controller for playing games on a large public display. *International Journal of Computer Games Technology*, 2008. 49

[VES09] VESA. VESA Net2Display remoting standard version 1. https://vesa.sharedwork.com/download/docid/8533011/view/N2Dv1.pdf [Last accessed: May 2014], October 2009. 81

[VHP+07] Johannes Vetter, John Hamard, Massimo Paolucci, Enrico Rukzio, and Albrecht Schmidt. Physical mobile interaction with dynamic physical objects. In *Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '07, pages 339–340, New York, NY, USA, 2007. ACM. 43

[VO11] Ville Valkama and Timo Ojala. Stakeholder value propositions on open community testbed of interactive public displays. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 107–113, New York, NY, USA, 2011. ACM. 57, 74

[W3C03] W3C. Synchronized multimedia. http://www.w3.org/AudioVideo/ [Last accessed: May 2014], 1998-2003. 61, 166

[W3C04] W3C. XML schema part 0: Primer second edition. http://www.w3.org/TR/xmlschema-0/ [Last accessed: October 2013], 2004. 205

[W3C08] W3C. Offline web applications: W3c working group note. Technical Report 30 May 2008, W3C, 2008. 156, 160

[W3C13] W3C. SPARQL 1.1 overview. http://www.w3.org/TR/sparql11-overview/ [Last accessed: November 2013], March 2013. 228

[WB97] Mark Weiser and John Seely Brown. The coming age of calm technology. In Peter J. Denning and Robert M. Metcalfe, editors, *Beyond Calculation*, pages 75–85. Copernicus, New York, NY, USA, 1997. 21

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific American*, September 1991. 19, 21

[WFC06] Amanda Williams, Shelly D. Farnham, and Scott Counts. Exploring wearable ambient displays for social awareness. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '06, pages 1529–1534, New York, NY, USA, April 2006. ACM. 21

[WGF06] Stephanie Wilson, Julia Galliers, and James Fone. Not all sharing is equal: The impact of a large display on small group collaborative work. In *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, CSCW '06, pages 25–28, New York, NY, USA, 2006. ACM. 33

[WHaG92] Roy Want, Andy Hopper, Veronica Falc ao, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992. 27, 82

[WHCS08] Adam Wolbach, Jan Harkes, Srinivas Chellappa, and Mahadev Satyanarayanan. Transient customization of mobile computing infrastructure. In *Proceedings of the First Workshop on Virtualization in Mobile Computing*, MobiVirt '08, New York, NY, USA, June 2008. ACM. 53, 89, 91, 99, 191

[Wi-12] Wi-Fi Alliance. Wi-Fi CERTIFIED Miracast^{TM}. `http://www.wi-fi.org/miracast` [Last accessed: October 2012], 2012. 54

[WID+98] Craig Wisneski, Hiroshi Ishii, Andrew Dahley, Matthew G. Gorbet, Scott Brave, Brygg Ullmer, and Paul Yarin. Ambient displays: Turning architectural space into an interface between people and digital information. In *Proceedings of the First International Workshop on Cooperative Buildings, Integrating Information, Organization, and Archi-*

tecture, CoBuild '98, pages 22–32, London, UK, 1998. Springer-Verlag. 22

[WPD+02] Roy Want, Trevor Pering, Gunner Danneels, Muthu Kumar, Murali Sundar, and John Light. The personal server: Changing the way we think about ubiquitous computing. In *Proceedings of the 4th International Conference on Ubiquitous Computing*, Ubicomp '02, pages 194–209, London, UK, 2002. Springer-Verlag. 53

[WRB+97] Kenneth R. Wood, Tristan Richardson, Frazer Bennett, Andy Harter, and Andy Hopper. Global teleporting with Java: Toward ubiquitous personalized computing. *Computer, IEEE*, 30(2):53–59, February 1997. 82

[WT13] The WebKitGTK+ Team. The WebKitGTK+ project. http://webkitgtk.org/ [Last accessed: November 2013], 2009–2013. 196

[XCDS13] Yu Xiao, Sarah Clinch, Nigel Davies, and Mahadev Satyanarayanan. Move closer : The benefits of a flexible approach to display and application placement (demo). In *Proceedings of the 2013 International Symposium on Pervasive Displays*, PerDis '13. ACM, 2013. 237, 239, 291, 293, 295

[Yev11] Maksim Yevmenkin. l2ping. http://www.freebsd.org/cgi/man.cgi?query=l2ping [Last accessed: March 2013], March 2011. 60

[Yiu13] Otto Yiu. django-cors-headers. https://github.com/ottoyiu/django-cors-headers [Last accessed: October 2013], 2013. 215

[YTH+05] Koji Yatani, Koiti Tamura, Keiichi Hiroki, Masanori Sugimoto, and Hiromichi Hashizume. Toss-it: Intuitive information transfer techniques for mobile devices. In *Extended Abstracts on Human Factors in Computing Systems*, CHI '05, pages 1881–1884, New York, NY, USA, April 2005. ACM. 48

[YTH$^+$06] Koji Yatani, Koiti Tamura, Keiichi Hiroki, Masanori Sugimoto, and Hiromichi Hashizume. Toss-it: Intuitive information transfer techniques for mobile devices using toss and swing actions. *IEICE Transactions on Systems and Computers*, E89-D(1):150–157, 2006. 48, 104

[YTSH04] Koji Yatani, Koiti Tamura, Masanori Sugimoto, and Hiromichi Hashizume. Information transfer techniques for mobile devices by toss and swing actions. In *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*, WMSCA 2004, pages 144–151, 2004. 48

[Zen13] Zend Technologies Ltd. Zend framework. `http://framework.zend.com/` [Last accessed: November 2013], 2006–2013. 226, 228

[Zet14] Zetakey Solutions. Zetakey HTML5 webkit browser. `http://www.zetakey.com/` [Last accessed: May 2014], 2009–2014. 61