Chapter XIII
A Tool Supported Methodology For Developing Secure Mobile P2P Systems

James Walkerdine
Lancaster University, UK

Peter Phillips
Lancaster University, UK

Simon Lock
Lancaster University, UK

ABSTRACT

The growth of mobile devices with near PC equivalent capabilities has brought with it the possibility of mobile Peer-to-Peer (P2P) systems. However, the unique nature of mobile devices introduces new challenges that need to be considered during the development process, especially when considering critical aspects such as system security. This chapter presents the PEPERS Development Methodology (PDM), a tool-supported methodology that aims to assist designers in developing secure mobile P2P systems, and encourages them to consider specific mobile P2P design issues from an early stage. The PDM is demonstrated within the context of a real-world case study of a system developed for a security company.

INTRODUCTION

The rapid growth in availability of mobile devices with ever increasing functionality has brought with it the possibility of mobile Peer-to-Peer (P2P) systems. More recently, advances in wireless networking and mobile computing technologies, such as wireless LANs, wireless mesh networks and 3G cellular networks have further facilitated the migration of the P2P paradigm into wireless mobile computing. The combination of mobile and P2P technologies could be ideal for organisations that possess characteristics such as, decentralised management styles, geographically dispersed or highly mobile workforces, a wide range of computing and communications devices, etc.

However in developing real-world mobile P2P systems, designers can face a new set of development challenges - particularly when it comes to providing security and privacy. Ensuring such characteristics exist within a system is of particular importance in an environment that, by its very nature, is ad-hoc and heterogeneous.

The issue is complicated, however, by the fact that the P2P approaches used and the underlying mobile technologies will also have an impact on a system's characteristics (Walkerdine, 2001). For example, the choice of P2P topology can significantly impact on how well a system can provide security. Decentralised P2P systems are likely to be better suited for handling denial of service attacks; the central authority provided by semi-centralised P2P systems would be better suited for handling authentication. Likewise the resource constraints (e.g. memory, battery life) of mobile devices can limit the amount of computation that can be performed or the level of communication between devices.

Such issues are important, and necessary to consider when developing a secure mobile P2P system. Having a development method to accommodate them is vital but, unfortunately, is something which is currently lacking within the domain. This chapter presents the PEPERS Development Methodology (PDM), a tool-supported methodology that provides such development assistance. The PDM encourages designers to consider the issues that are central to mobile P2P system development (for example, identifying security concerns, considering mobility and technical constraints, and making architectural design decisions) from an early stage. It supports the designer throughout the development cycle, from initial requirements elicitation through to the final implementation - ensuring that security and mobile concerns are properly addressed throughout. In particular, it assists developers in determining the most suitable P2P topologies and application reference architectures for their design, based on the system, security and mobility requirements that they have identified. The PDM comes with tool support and this is also illustrated within this chapter.

The PDM was developed as part of the EU funded PEPERS (PEPERS, 2006) project that has developed an infrastructure to support the design, development and operational deployment of secure mobile P2P applications. The outcomes of the project have been utilised and evaluated by industrial user partners within their own business domains, as well as a number of mobile software development companies. In order to help illustrate the use of the PDM and support tool, a real-life case study is provided that involves one of these industrial systems.

This chapter begins by discussing some of the key issues that need to be considered when developing a secure mobile P2P system. An overview of the PDM is then provided, along with a description of the case study and developed tool support. The PDM is then described in detail, referring to the case study to demonstrate each stage in use.

BACKGROUND

Although in general developers can draw upon existing software engineering techniques for developing secure mobile P2P systems, the distributed and unpredictable nature of P2P coupled with the technical challenges of mobile technology and security constraints, means that there are a number of specific issues that designers must consider.

Making Security Central to the Design

Successful security functionality is not something that can be bolted onto a system as an afterthought. The broad nature of security means that it is something that should be considered at all stages of a systems development. For example, technical and company policy issues can have an impact on proposed security requirements; the choice of P2P topology can impact on how these requirements are met; the 3rd party encryption component that is to be used can likewise have an impact, etc. It is therefore important to make security a central concern throughout the development lifecycle. Such activities can also be supported from an organisational perspective by adopting a rigid review process, and assigning members of the development team specific roles to ensure that security requirements are being properly considered.

Mobility Requirements and Constraints

Mobile applications and mobile technologies introduce new sets of requirements and constraints. For example, the degree of mobility that is expected within an application will impact on how it is designed (from a P2P perspective high mobility might suggest a more decentralised approach). In addition, the technology capabilities of a mobile device can restrict memory availability (and thus program size), or the amount of processing the device can perform. Likewise mobile technology will introduce its own requirements such as battery life or those derived from restrictions imposed by the devices OS. Given the influence mobility can have on a systems design, it is important for developers to consider it at all stages of the development lifecycle and, where practical, for mobile technology decisions to be finalised as early as possible.

Network and Communication Requirements and Constraints

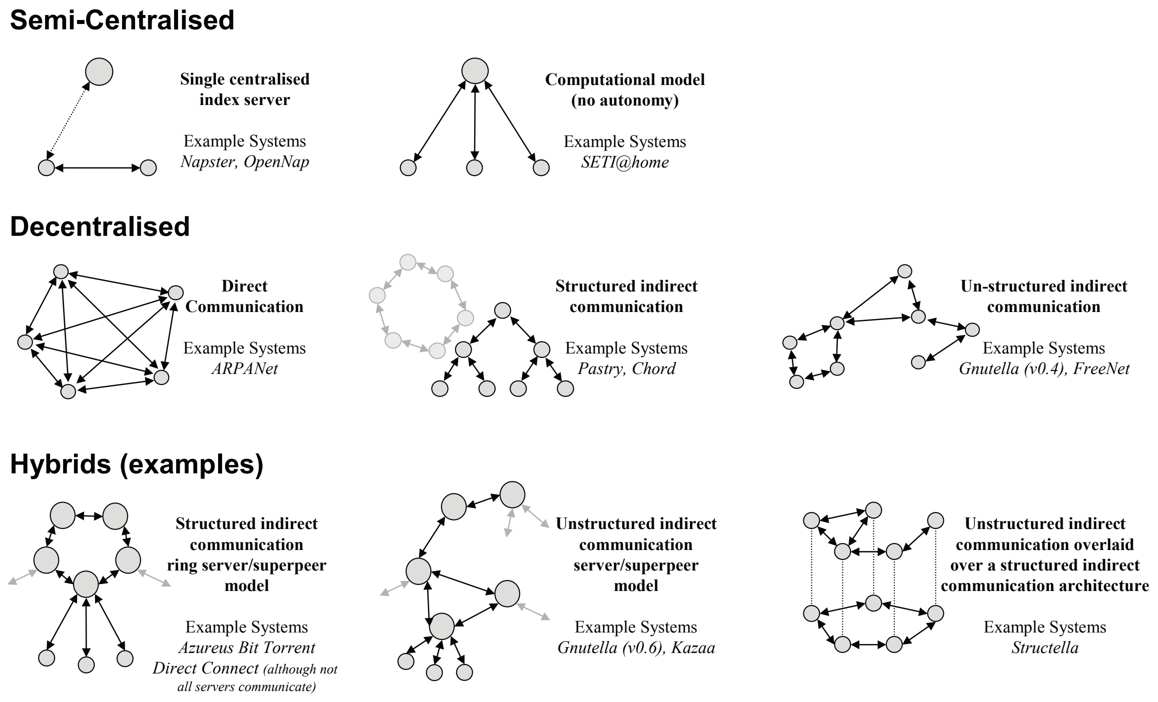
The carrier or network type, and the expected network coverage level can all have implications on a developed system. Of particular importance is the communication mechanism that will be used, with various tradeoffs that exist between bandwidth capabilities, range capabilities and power consumption characteristics. During development designers will need to consider the characteristics/requirements of their system and how these relate to the functionality provided by the available technology. This will also influence other design decisions, for example the type of P2P topology that should be used.

P2P Technology Requirements and Constraints

An important part of P2P software development is the definition and analysis of the underlying topology that is to be used. Our previous work (Walkerdine, 2002) has shown that the choice of topology can have a significant impact on the properties (e.g. security) of a system, perhaps most clearly highlighted with the differences between decentralised and semi-centralised architectures. It is important for designers to realise and understand the effect that the choice of topology can have on a systems specification and design (and likewise how a system's requirements can influence the choice of topology). For context, Figure 1 presents a summary of the key classes of P2P topologies.

The choice of P2P technology (for example, the protocol and implementations used) can also have an influence, with each typically offering varying degrees of functionality to the developer. For example, JXTA provides considerably more P2P development support than .NET, and consequently less needs to be provided within any system design. Likewise, the distributed P2P security implementations that are provided

Figure 1. Key classes of P2P topologies


Semi-Centralised
Single centralised index server
Example Systems
Napster, OpenNap
Computational model (no autonomy)
Example Systems
SETI@home
Decentralised
Direct Communication
Example Systems
ARPANet
Structured indirect communication
Example Systems
Pastry, Chord
Un-structured indirect communication
Example Systems
Gnutella (v0.4), FreeNet
Hybrids (examples)
Structured indirect communication ring server/superpeer model
Example Systems
Azureus Bit Torrent
Direct Connect (although not all servers communicate)
Unstructured indirect communication server/superpeer model
Example Systems
Gnutella (v0.6), Kazaa
Unstructured indirect communication overlaid over a structured indirect communication architecture
Example Systems
Structella

by (Berket et al., 2004) and (Agarwal et al., 2001), each offers differing levels of security support. Given this, it will be important for developers to commit to a technology at an early stage within the development lifecycle.

Adopting an Architectural Driven Design Approach

Due to the fact that architectures play a key role in P2P development, it is beneficial to use a design approach that places a greater emphasis on these issues. Architectural driven design approaches specifically encourage the development of the system architecture from early within the development lifecycle, and so represent an ideal option for supporting P2P system development. Such approaches also support the use of reference architectures that can act as inputs to the design process - helping developers determining the architectural building blocks they might need.

THE PEPERS DEVELOPMENT METHODOLOGY

The PEPERS Development Methodology (PDM) was originally conceived in the BANKSEC project (BANKSEC, 2000), in which a methodology was needed for specifying system and component security requirements. During P2P ARCHITECT (P2P ARCHITECT, 2001), this methodology was further developed and tailored for use in the specification and design of dependable P2P software systems. Within PEPERS it has been further refined to support the development of secure mobile P2P applications. Within each project the methodology has been used and evaluated by end user industrial partners. As a consequence, the PDM is a result of many years of practical industrial use and theoretical refinement (it has been successfully applied in the development of banking and P2P based online booking systems).

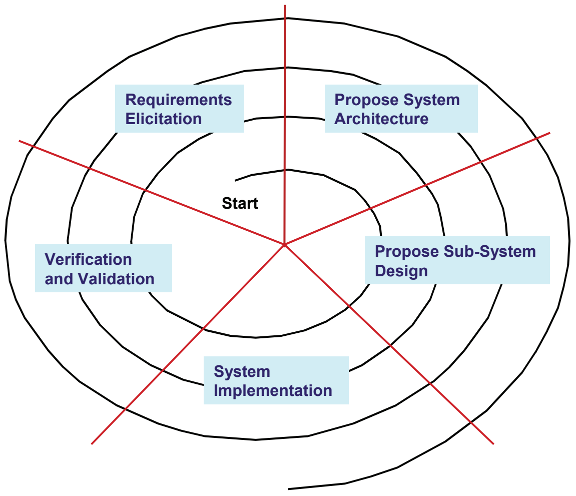
Existing methodologies already exist for more general mobile application development (for example the UML based development approach described in (B'Far, 2004) and the work on network-based software architectures in (Fielding, 2000)). Middleware solutions, such as PROEM (Kortuem, 2002) have also been developed to provide a foundation for the development of mobile P2P systems. In addition there has also been work in developing more innovative methods, such as ontology-based development methods, e.g. Tropos (Bertolini, 2002; Mouratidis, 2003), to aid P2P system design. However, no general software engineering methodology exists that particularly encourages developers to consider security and P2P issues, and thus in this regard, the PDM is novel.

The PDM is based on a 5-stage spiral model as illustrated in Figure 2. This spiral model is comprised of five segments:

- *Requirements Elicitation*
- *Propose P2P system architecture*
- *Propose sub-system design*
- *System Implementation*
- *Verification and Validation*

The spiral nature of the PDM provides the advantage of a flexible development process in which designers are not rigidly constrained by fixed phases. An important characteristic of the method is that it is *iterative*, in that during the design of a P2P application the designers can revisit stages of the model as new requirements, architectural issues, etc, come to light. It is likely, for example, that early on in a P2P applications development, iterations of the model will largely focus on the gathering and defining of requirements, and later stages on sub-system design. This spiral and iterative nature of the process is important for accommodating the fact that a P2P system's design will often change during development. This may be as a result of a range of factors, including the resolution of requirement conflicts, the late identification of additional requirements, changes in the application domain or business context and so on.

Figure 2. The stages that comprise the PDM


Requirements Elicitation
Propose System Architecture
Start
Verification and Validation
Propose Sub-System Design
System Implementation

The PDM is flexible enough to accommodate different types of software engineering techniques during development. For example, different requirement elicitation techniques, or design approaches.

A detailed breakdown of the PDM is provided later in this chapter. To help illustrate this and the use of the support tool a real-world case study is also presented. To provide some background, the chapter will first provide an overview of the case study and the supporting tool.

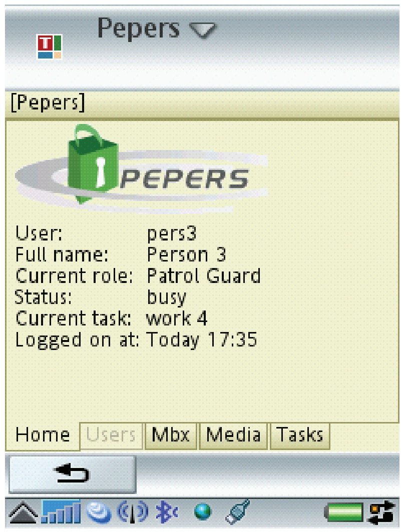
Case Study: Supporting a Security Firm

To help in the design and evaluation of the methods and tools developed within PEPERS, the project consortium includes industrial user partners who wish to use P2P technology to support their own business practices. One of these is a major international security firm who wishes to support their guards in communicating, collaborating and sharing information with each other whilst on patrol. Another is a media company who wishes to allow their journalists, photographers and editors to work together in the creation of magazines and newspaper articles. In both cases the companies wish to use P2P to support secure communication and data exchange between their personnel, who may be geographically dispersed or particularly mobile, and may only have access to mobile devices. Within the project each user partner has built a secure mobile P2P system that specifically supports elements of their business processes. The PDM was used as the underlying methodology for the development of the applications in both cases.

For this case study we will focus on the pilot that was developed with the security firm. Its aim was to allow its security guards and mobile patrols to receive and transmit sensitive customer information in a dynamic environment via their mobile devices. Such environments are typically ad-hoc exceptional situations which the guards/patrols have responded to and which involves them co-operating with one another. Guards are initially assigned tasks via the ARC, a central computer based in the company's headquarters.

The pilot was built for operation on a Sony Ericsson Symbian 9 OS phone, and has been successfully deployed and evaluated within the organisation for a period of 6 months. Figure 3 provides a screenshot of the developed security pilot application running on the phone.

Figure 3. The Security firm pilot



User: pers3
Full name: Person 3
Current role: Patrol Guard
Status: busy
Current task: work 4
Logged on at: Today 17:35

Using the application guards are able to receive tasks from the ARC, update it on their progress, as well as communicate messages and images directly with each other. All communication is encrypted, and repeated authentication is required to access the applications functionality. Dynamic verification techniques are also used to ensure that the user does not perform actions that are inappropriate for their role.

Tool Support for the PDM

In order to support designers in using the PDM, web based tool support was also built. Designed to be generic in nature, it is able to work alongside different system engineering paradigms (for example, architectural driven or component based design). Not only does the tool support the development of systems in accordance with the PDM, it also actively encourages the consideration of security issues by the developers.

The tool itself can play a part during the first three stages of the method. It provides support for requirement gathering and analysis, as well as assisting the designers in making informed decisions with regards to the architectural design. Designers are able to draw upon a knowledge base of topologies and P2P application reference architectures, and the characteristics (i.e. the effect on system properties, etc). They are also able to identify and provide initial descriptions for the sub-systems within their design. The intention is that designers use separate tools (for example, Rational Rose) to then fully develop the system architecture and design.

The tool itself provides a number of key features for designers:

- *Support for identifying, specifying and managing requirements*. The tool can assist developers in identifying and specifying system requirements during software development.

- *Support for topology selection.* The tool draws upon a knowledge base of P2P topologies and details about the impact they may have on a range of security and mobility properties. The tool is able to present this knowledge to the developers enabling them to make informed decisions, as well as being able to suggest suitable topologies.
- *Support for the identification of key secure mobile P2P application functionality.* The tool assists the designer in identifying key areas of desired application functionality (functional capabilities). This could be, for example, encryption or instant messenger capabilities.
- *Support for secure mobile P2P Application Reference Architecture selection*. The tool is able to draw upon a knowledge base of application reference architectures and the capabilities they typically support. The tool can present this knowledge to the designers and suggest application reference architectures that may be of relevance.
- *Support for Sub-system identification and initial description*. The tool provides facilities to allow designers to identify and provide initial descriptions for the sub-systems used within their design. These can be outputted and further developed within separate design packages.
- *Support for general managerial and trace ability activities*. The tool provides a number of managerial features including report generation and traceability support, allowing designers to trace their various decisions throughout the design process. For example, by linking the choice of topology to the set of requirements that motivated this design decision. The output of the tool can also be imported into UML tools (via XMI).

BREAKDOWN OF THE PDM

The chapter will now move on to describe the five stages of the PDM in detail. The case study will be visited at each stage to help illustrate the PDM in use. It should be noted that although the user partner performed many iterations of the PDM's spiral before the design was complete; for the sake of brevity these are not all fully documented here.

Requirements Elicitation

Before commencing this stage, a business decision to procure a P2P system to support some business activity must first have been made. From this decision, a number of business requirements will be generated. These business requirements set out the critical functionality that the P2P system is expected to provide and define the constraints on the operation of the system that is to be developed.

From these high level goals, more concrete system requirements are generated and specified. Part of this will involve the identification of security requirements many of which will be applicable to the P2P system as a whole rather than to the individual parts that comprise it. In summary, activities in this stage of development would include:

- *Elicitation of business goals and requirements from stakeholders*. The identification of mobile and security requirements should play a central role in the elicitation process. Part of this should involve a detailed security analysis, using techniques such as threat modelling to help identify the security risks and the (differing) levels of security that will exist within the system. Stakeholders should also be encouraged to identify the mobility attributes they desire for the system.
- *Elicitation system requirements from stakeholders and business requirements*. Again

mobile and security aspects should be an important focus.
- *Analysis and negotiation of agreed system requirements*. Negotiate any conflicts between stakeholder requirements.
- *Production of system requirements document*.

The properties of the various P2P topologies and the effect that these can have on system requirements will play a significant part at this stage of development. For example, identified requirements may restrict the choice of topology (for example, a system that is to be highly managed would benefit most from a semi-centralised topology). Conversely the selection of a topology may produce additional functional requirements for the system (for example, the need for super peers). In either case, there is likely to be conflicting issues between the requirements and the topologies that would need to be resolved.

Case Study

During the initial iterations of the model in this stage the user partner identified a number of business and security requirements for their P2P system (a small selection of which are as follows):

- Mechanisms should be in place to verify the identity of users when taking on specific roles within the system (e.g. ARC operator, Mobile Unit member, Team Leader etc)
- The system should manage data on a strict need-to-know basis
- It is essential that location, status, activity, assignment, incident and project data does not become available to unauthorised individuals
- The system should not require additional hardware expenditure of more than 5000 euros

The user partner then proceeded to identify a set of initial viewpoints and system requirements for their P2P system, using the tool support to help document them. As the designers gained a clearer understanding of the different stakeholders needs, the system requirements were revised and specified in more detail. Identified viewpoints and requirements included:

- **Shift Supervisor:** The person in change of assigning shifts, briefing and de-briefing guards
- **Shift Creation:** The system should allow the Shift Supervisor to create and describe shifts (including relevant security data)
- **Guard Monitoring:** The system should be able to log all the activities of a mobile unit (guard) including all communications, actions on data, changes in status and movements in the real world
- **Guard:** The person who takes part in security related activities
- **Information Request:** The guard should be able to request information from the system that is required in order to complete their task.

As the specification and design developed, the designers were able to use the support tool to begin identifying which security properties and system capabilities where relevant. In turn the tool was able to suggest topology and P2P application reference architectures that may be suitable for their design (as described below).

Propose System Architecture

The architecture of any system is the framework that defines how the entities within it (e.g. components, objects, etc) are organised and what relationships may exist between them.

Before developing the architecture it is usually necessary for at least some requirements to have been defined. However a complete set of requirements will usually not be available in the very early stages of the development. Initial requirements can be used to generate preliminary ideas of the architectural design, and by defining architectures early on it can provide the designers with ways of organising the process of specification and design. From a secure mobile P2P perspective, however, it is important that initial security and mobility requirements have been identified and considered, before this stage is begun.

Furthermore, when developing mobile P2P systems it is highly likely that the specification and the design will be interwoven, with each affecting the other as system development proceeds. The utilisation of an iterative approach allows these two to be developed in concert with each other.

The activities designers of mobile P2P systems would carry out in this development phase include:

- *Select topologies*. Based on the identified system requirements, and mobile and security properties for the P2P system, designers can filter and select the appropriate topologies. Knowledge of the different topologies and their influence on system properties (Walkerdine, 2002; Walkerdine, 2006) can be used as an input into this activity. For example, semi-centralised topologies can assist in the provision of authentication and authorisation functionality. However, the centralised points they possess also act as points of weakness in the architecture – making them susceptible to attacks.
- *Derive system functional capabilities*. Designers should derive from the identified requirements, system functional capabilities. These capabilities represent abstract system functionality that the system should possess. For example, it may be desired for the system to possess Instant Messaging capabilities, or encryption capabilities. The identification of these capabilities can help designers in selecting relevant secure mobile P2P application reference architectures.
- *Select mobile P2P application reference architectures*. Designers can use the capabilities that have been identified in the previous activity to help select secure mobile P2P application reference architectures that could be used as input into the design of their system. Within PEPERS a set of reference architectures have been developed that reflect key application domains within the field (e.g. shared workspace, distributed storage, etc) and the different types of P2P topology (Walkerdine, 2005).
- *Establish architectural model*. This is where an overall model of the system at a high level is developed; it shows the system entities and the relationships between them. Designers can draw upon the application reference architectures and their capabilities to help them gain an understanding of what may be required, and also to provide references to compare their own designs against. Furthermore, the choice of topology and identified requirements will also have an effect on the architectural model (for example, whether it is to be a decentralised or semi-centralised system). Identified security and mobility requirements will play a key role within this activity, impacting on the architectural model and vice versa - and so should be considered carefully.
- *Describe sub-systems*. Here the functionality of each sub-system is defined and attention is drawn to sub-system boundaries. Again the application reference architectures can be used to help the designers to initially think about the sub-systems, the services they provide and their boundaries. Likewise the identified requirements will be an important input to this activity.
- *Where possible, allocate requirements to sub-systems*. Requirements that have been identified and specified for the system should be mapped on to the architecture and its sub-systems. Ideally, each requirement

will be assigned to a single sub-system. In practice, however, this is rarely possible as a requirement may encompass several sub-systems (particularly with mobility and security requirements).

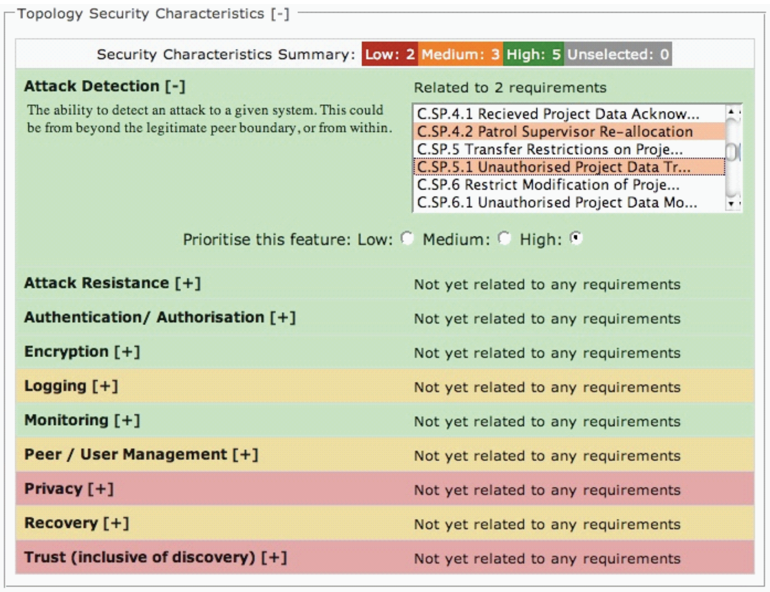
- *Evaluate architecture*. This involves checking to ensure that all requirements have been assigned to sub-systems, looking for mismatches between the architecture and the requirements and assessing what problems may arise when implementing the architecture. Assessing how well the security and mobile technology requirements have been satisfied will be particularly important.

Case Study

Having performed initial requirements elicitation activities, the designers began to investigate the suitability of the different topologies and application reference architectures for use in their architectural design.

As shown in Figure 4, the designers used the supporting tool to assign ratings to a range of security properties (top diagram). For each property the developers indicated its importance to their design and to what requirements it related to. Based on this input the tool was able to suggest that a semi-centralised topology would be the most suit-

Figure 4. Rating the importance of security properties, and using these to suggest suitable topologies



Topology Security Characteristics [-]

Security Characteristics Summary: Low: 2 Medium: 3 High: 5 Unselected: 0

Attack Detection [-]

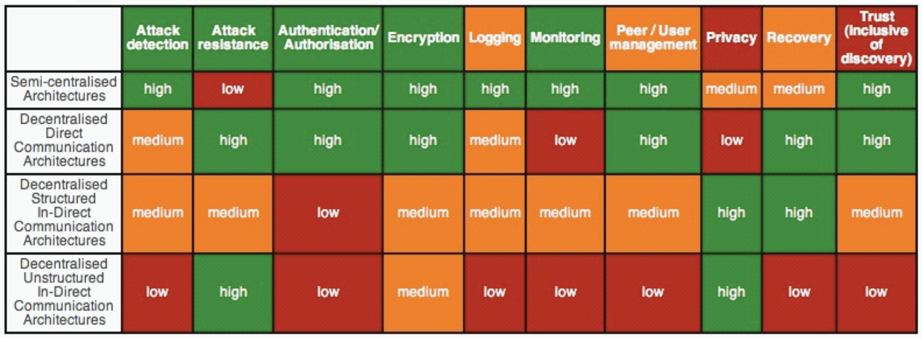
The ability to detect an attack to a given system. This could be from beyond the legitimate peer boundary, or from within.

Related to 2 requirements

- C.SP.4.1 Recieved Project Data Acknow...
- C.SP.4.2 Patrol Supervisor Re-allocation
- C.SP.5 Transfer Restrictions on Proje...
- C.SP.5.1 Unauthorised Project Data Tr...
- C.SP.6 Restrict Modification of Proje...
- C.SP.6.1 Unauthorised Project Data Mo...

Prioritise this feature: Low: ○ Medium: ○ High: ●

Attack Resistance [+]	Not yet related to any requirements
Authentication/ Authorisation [+]	Not yet related to any requirements
Encryption [+]	Not yet related to any requirements
Logging [+]	Not yet related to any requirements
Monitoring [+]	Not yet related to any requirements
Peer / User Management [+]	Not yet related to any requirements
Privacy [+]	Not yet related to any requirements
Recovery [+]	Not yet related to any requirements
Trust (inclusive of discovery) [+]	Not yet related to any requirements



	Attack detection	Attack resistance	Authentication/ Authorisation	Encryption	Logging	Monitoring	Peer / User management	Privacy	Recovery	Trust (inclusive of discovery)
Semi-centralised Architectures	high	low	high	high	high	high	high	medium	medium	high
Decentralised Direct Communication Architectures	medium	high	high	high	medium	low	high	low	high	high
Decentralised Structured In-Direct Communication Architectures	medium	medium	low	medium	medium	medium	medium	high	high	medium
Decentralised Unstructured In-Direct Communication Architectures	low	high	low	medium	low	low	low	high	low	low

able for meeting their requirements. The bottom diagram shows the topology recommendations based on the desired security properties, with the most suitable at the top. In both diagrams, green represents high priority, orange for medium priority and red for low priority.

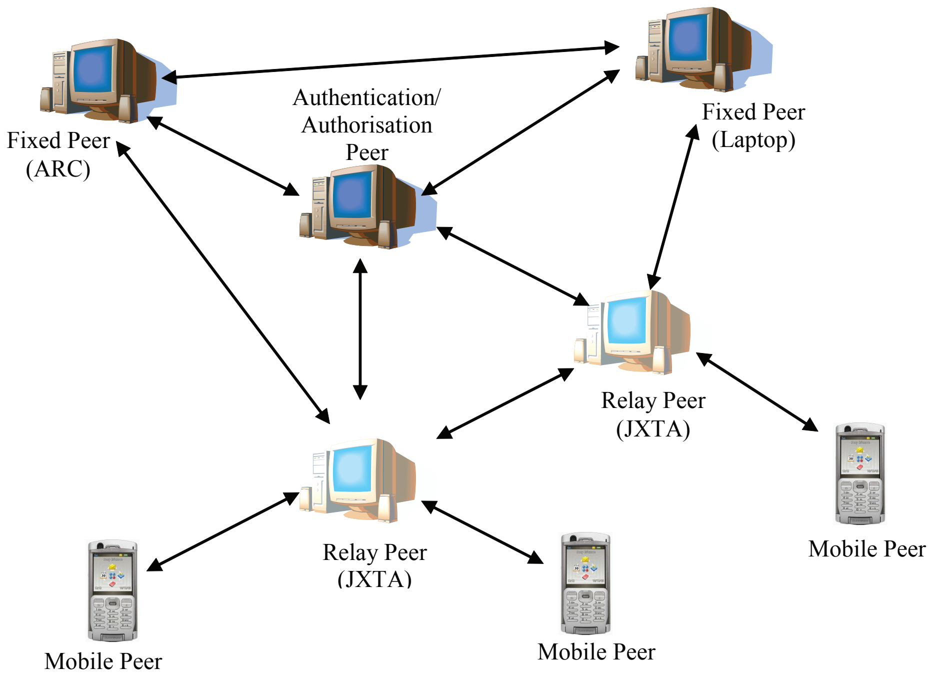
Upon further analysis of the system requirements (in particular the fact that the existing system possesses a single centralised server - the ARC), a design decision was made to use a semi-centralised topology as a basis for the system. Because the users existing system already possessed a central server, the choice of topology did not impose any additional requirements/constraints onto the design that had not already been considered. Figure 5 provides a general overview of the topology used within the system.

The designers also made use of the support tool in order to gain suggestions for secure mobile P2P application reference architectures that could provide an input to their design. Although none of the reference architectures known by the tool were directly relevant to the system being developed, the Instant Messenger and Shared Workspace reference architectures were found to be the closest fit.

At this time the designers also began to develop the architectural model for their system. The topology was the foundation of the system design, and this was extended so that the structure of the different peer types (standard or super peers) that comprised it, were then specified. As part of this process, the designers drew upon the reference architectures, and the requirements document.

From studying the reference architectures the designers identified the key sub-systems they would require in their architectural design. Not all of the sub-systems described within the reference architectures were used, with the designers feeling that the functionality relating to real-time connection monitoring and logging was unnecessary in their pilot. Likewise the designers decided that limited awareness functionality would be incorporated within the P2P Communication sub-system, rather than as a standalone sub-system as proposed within the reference architectures. Using

Figure 5. General overview of the topology used within the developed system

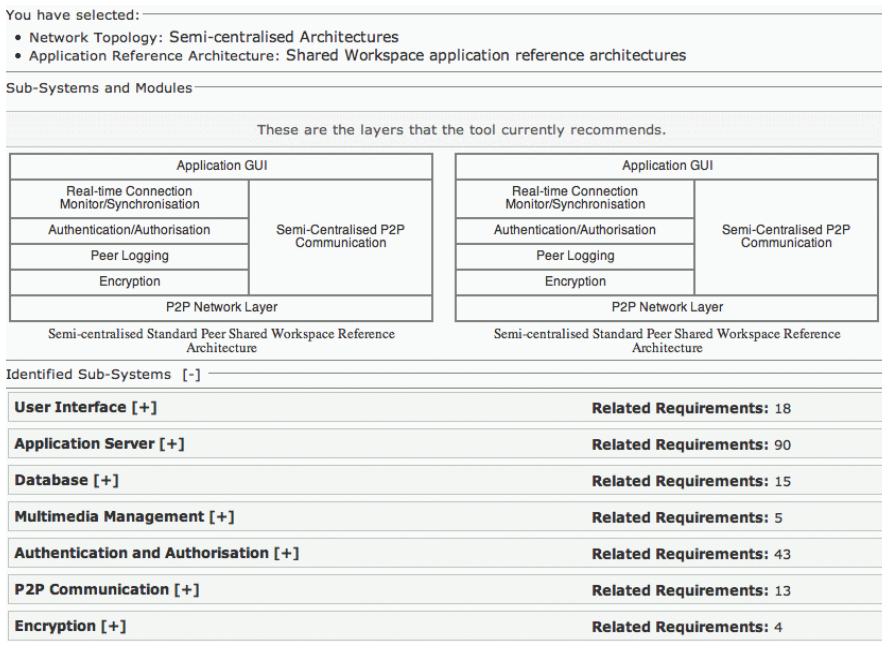


the support tool, the designers began to briefly describe the sub-systems they had identified for their system. Figure 6 provides a screenshot of the tool showing topology and reference architecture choices, and the sub-systems that have been identified. Using a separate design tool, these sub-systems were then organised into high-level architectures for the peer (Figure 7), before being developed into a full design.

Propose Sub-System Design

This stage focuses on the actual design of the P2P system; taking the proposed system architecture

Figure 6. Reviewing choices and identifying sub-systems


You have selected:

- Network Topology: Semi-centralised Architectures
- Application Reference Architecture: Shared Workspace application reference architectures

Sub-Systems and Modules

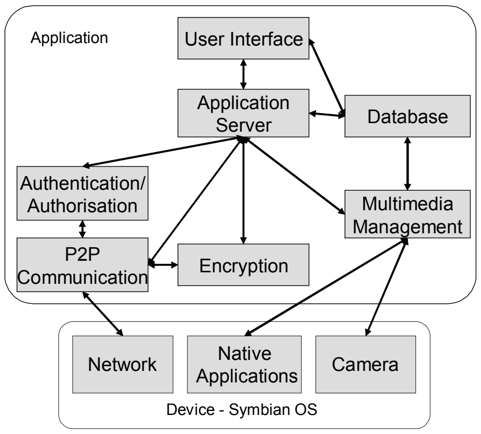
These are the layers that the tool currently recommends.

Application GUI
Real-time Connection Monitor/Synchronisation
Authentication/Authorisation
Peer Logging
Encryption
Semi-Centralised P2P Communication
P2P Network Layer
Semi-centralised Standard Peer Shared Workspace Reference Architecture
Application GUI
Real-time Connection Monitor/Synchronisation
Authentication/Authorisation
Peer Logging
Encryption
Semi-Centralised P2P Communication
P2P Network Layer
Semi-centralised Standard Peer Shared Workspace Reference Architecture

Identified Sub-Systems [-]

User Interface [+]	**Related Requirements:** 18
Application Server [+]	**Related Requirements:** 90
Database [+]	**Related Requirements:** 15
Multimedia Management [+]	**Related Requirements:** 5
Authentication and Authorisation [+]	**Related Requirements:** 43
P2P Communication [+]	**Related Requirements:** 13
Encryption [+]	**Related Requirements:** 4

Figure 7. High level architecture for a standard peer



and requirements specification, and using these to create a detailed design for the system. To do this an initial architecture for the system must have been developed.

The activities the designers would carry out in this stage include:

- Specifying the entities that make up the system architecture and its subsystems (objects, components, etc)
- Specifying the relationships that may exist between these entities.
- Specifying the allocation of system requirements to these entities.
- Specify, if necessary, the non-functional attributes that an entity should have

As part of PEPERS a set of runtime modules have been developed that provide general functionality (for example, encryption, secure P2P communication, etc) intended for use by mobile P2P applications (Walkerdine, 2007). When designing the various application sub-systems, developers may wish to make use of these modules. The tool support has knowledge of such modules and can direct the developer towards using them during the design process.

Initial implementation choices will also impact on the design stage. Because a single development standard does not exist for P2P systems, a design can change significantly depending on the underlying technology that is to be used. Likewise the choice of mobile technology will also have repercussions on the system design. To support the design process, initial implementation choices should be made early, with the PDM's design and implementation stages re-visited as the development progresses.

Case Study

By the time the user partner focused their efforts on this stage, the requirements and application architecture had been through a number of iterations of development and some initial decisions had been made with regards to implementation technology. The design progressed with reference to the Shared Workspace reference architecture suggested by the support tool. The designers proceeded to specify the design of the various sub-systems in detail, using a separate design tool (Rational Rose). For those they intended to develop themselves, this involved breaking them down into components, which were then further expanded with class definitions. The individual classes were then specified, with attributes and methods defined.

The designers decided that, where possible, the sub-systems would be based around those provided by the PEPERS runtime modules. In particular their design would draw upon P2P Communications, Authentication/Authorisation and Encryption modules. The designers integrated the interfaces for these modules into their design.

The design was also influenced by the implementation technology decisions that had been made. In particular it was decided that a Sony Ericsson phone would be used and the application would be developed in Java. This in turn introduced requirements that fed back into the requirements elicitation and placed limitations on the architecture/design stages.

System Implementation

This stage takes the requirements, high-level architecture and detailed design from the previous stages and uses them to guide the actual construction of the final system. Initial iterations of this stage will typically focus on determining the types of technology that would be used as part of the implementation. The PDM does not prescribe set implementation approaches and instead allows developers and organisations to use the processes they are already familiar with.

The activities developers would carry out in this stage include:

- *Decide on P2P technologies*. For example, the P2P protocol and API
- *Decide on mobile technologies*. For example, the mobile device and communication technologies.
- *Decide on security technologies*. For example, encryption algorithms and related API's.
- *Implement system entities*. Using PEPERS runtime module implementations if applicable.
- *Integrate system entities*.

Designers should carefully consider the impact technology choices will have on their development (in terms of requirements/constraints). In many cases legacy technologies will also play a role within the system, especially if the system is to integrate with a clients existing setup (which typically will be more centralised and less mobile). Certainly during early stages of the development it will be important to re-visit the previous stages of the PDM, until the technical impact on the design has been fully considered.

Case Study

During early iterations of the development model, the user partners began to make decisions on how their system was to be implemented. It was decided that, where possible, the PEPERS runtime modules would be utilised and that the rest of the system would be built using Java. As a result of this decision, the bulk of the security functionality would be provided by the PEPERS modules. Additionally it was decided that a Sony Ericsson/Symbian 9 OS phone would be used as the basis of the mobile system hardware. The overall system would also need to integrate with the existing centralised system that formed the basis of the ARC central computer.

In later iterations, the focus moved to prototyping and implementing the sub-system designs developed during the previous stages. A key part of this was the integration of the PEPERS runtime modules with the rest of the components that had been developed in-house.

Verification and Validation

This stage focuses on the verification and validation of the system development and will occur at various points of the development lifecycle as the requirements, design and implementation are refined. Verification involves checking the system conforms to its specification. Validation involves checking the system as implemented meets the expectations of the client.

The activities carried out in this stage include:

- Validate the requirements against the stakeholders.
- *Validate the architecture against the requirements.* The architecture/requirements mapping performed in the Propose System Architecture stage will support this activity.
- *Validate the sub-system design against the architecture and requirements.* The requirement/ sub-system mapping performed as part of the Sub-system Design stage will support this activity.
- *Validate the implementation against the requirements/design*. Validating the implementation using testing techniques.
- *Verify the design models against the requirements.* Design models representing abstract system behaviour are verified against the requirements. Within PEPERS this activity is supported by the Static Verification Framework (SVF) (Siveroni, 2008).
- *Verify the implementation against the requirements.* The runtime execution of the system is verified against the requirements. Within PEPERS this activity is supported by the Dynamic Verification Framework (DVF) (Spanoudakis, 2008).

As a consequence of these activities, further iterations of the development model may be required so that additional requirements elicitation and negotiation, architecture and design refinement can be performed. As the PEPERS runtime platform modules have already been through a process of verification and validation, there use within a development can further help to support this process and help reduce the resources required for this stage.

Case Study

This stage was frequently visited during the development lifecycle. Initially the focus was on validating the identified business, security and system requirements with the clients. This was achieved through the use of sets of documented requirements (which were signed off by the user partners) as well as face to face requirement checking and negotiation workshops.

In later iterations of the methodology, validation moved on to focus on the architecture and sub-system design. During this phase of development, the user partners constructed semi-formal behavioural models of the system. These were then checked verified automatically using the SVF toolkit (Siveroni, 2008) to ensure the correct and expected operation of the system.

Finally the implementation was tested and debugged; drawing upon the DVF at runtime to make sure that incorrect usage behaviour was being blocked/reported.

EXPERIENCES IN USING THE PDM

As mentioned previously, industrial partners utilised the PDM to assist in the development of their pilot applications. This not only allowed the method and tool to be evaluated but also allowed refinements to be made based on feedback from industrial partner experiences. In addition, workshops were held with local mobile phone software companies to obtain additional third-party feedback. These companies were typically small in size, and so provided a different perspective to the software development process.

Overall the developers found the PDM and supporting tool to offer significant help in guiding the development of their secure mobile P2P applications. The smaller industrial companies were less sure about its use to them, mainly because they do not have the resources to follow a traditional development process and time to market is critical to them. They tended to use ‘extreme programming’ approaches to mobile software development. The larger companies, on the other hand, used more comprehensive development approaches and found the PDM to be of more relevance. This also highlights how the use of software engineering techniques within the real world is very much dependent on an organisations situation and its available resources.

The current development approaches they used were particularly linear in fashion (modelled on the ‘waterfall model’) and although it took time to get used to, they found the iterative nature of the PDM to benefit their way of working. The developers also found that the methodology and tool support was flexible enough to allow them to use existing processes and tools (such as their current UML design tools).

The developers found that the method encouraged them to consider specific security, mobile technology and P2P issues early within their systems development - something which their existing approaches did not achieve. Designers found the recommended secure mobile application reference architectures useful and that they were able to contribute to the development of a suitable architecture for their application.

However, despite these positive outcomes, due in part to the lack of experience with the method and the criticality of the application development, a number of issues were identified where it was felt the PDM could be improved.

Understanding the PDM

Many of the developers found the PDM to be significantly different from their current approaches to software development. As a result it took them a while to learn and gain an understanding of the steps that are involved. In particular the spiral, iterative approach was novel to them and it meant that their management style had to adapt to accommodate it. Given their inexperience with the method, there were times when this caused problems, resulting in detailed design decisions being made too early. When this occurred it became necessary for the designers to step back and re-assess their development. As the developers became more familiar with the methodology, however, such issues occurred less often. Given that all software development approaches take time to learn and understand, such findings are not that surprising.

Difficulty in Selecting Suitable Mobile P2P Application Reference Architectures

The developers initially found it difficult to identify which mobile P2P application reference architectures would be able to assist them in their design. The reference architectures themselves tackle specific types of application functionality, when in reality an actual system development may involve a combination of these functionalities. Initially the industrial partners tried to identify a single reference architecture that would satisfy all their requirements. It was only when they realised that this could not be achieved that they started to understand the true utility of these architectures; as points of reference for their own design. Supporting documentation that clarifies the use and benefit of reference architectures would encourage designers to make more appropriate use of them.

The Concept of Capabilities within the Reference Architectures

The developers had some difficulty in initially understanding the notion of a “capability” as used within the methodology and supporting tool. In particular their lack of experience of the P2P domain meant that they found it difficult to identify which capabilities would be relevant to their design. Such difficulties diminished as their experience grew, however it highlights the importance of novice P2P developers gaining a good understanding of the domain prior to development.

Improved Recommendation Support

Developers felt that that security property analysis provided by the tool could be made finer grained. Some of the security properties could be further broken down into sub-characteristics, reflecting the fact that developers may desire for a property to be provided for in different ways. For example, a systems attack resistance can be represented in different ways (lack of central points of failure, ability to resist tampering, etc). In addition, developers commented that the tool should provide a more detailed rationale behind its recommendations, further helping them to make their final architectural choices.

The Consideration of Other Non-Functional Properties

Although PEPERS has predominantly focused on mobility, security and P2P, the developers pointed out that for the PDM to be accepted as a development methodology it would also need to consider other non-functional properties such as reliability, scalability, etc. Although beyond the scope of the PEPERS project, expanding the PDM to consider such properties (or at least allowing it to draw upon suitable related work from areas such as dependability) is an area of future work.

CONCLUSION

There is an increasing interest in building mobile P2P applications. However for mobile P2P to be utilised within such an environment it also needs to be secure and the nature of P2P can make this difficult. A key problem is that the choice of P2P technologies can influence the properties of a systems design, and so consequently existing methodologies need to be extended or new ones developed in order to take this into account.

This chapter has presented the PEPERS Development Methodology, a method designed to assist in the development of secure mobile P2P systems. It adopts an architectural driven development approach and encourages developers to consider issues related to security, mobility and P2P from early within the development cycle.

The PDM was developed as part of the PEPERS project, one aspect of which involved its use within the development of real industrial P2P systems. To help in describing the PDM we have used one of these as a case study.

Experiences with using the PDM have shown that it can be a valuable aid in developing secure mobile P2P systems. In particular developers found that it made them more aware of architectural design issues and it encouraged them to think and consider them early on within their developments. There are a few areas where it could still be further refined. In particular the developers found the approach to be quite different from the more linear methods that they currently used, and the provision of clearer steps and more detailed examples would have helped in the quicker understanding of the method. The PDM also needs to consider other non-functional properties to widen its applicability.

Because we believe that tool support can play a crucial role within the development process, a tool has also been built that assists the designers in using the PDM. This tool has also been discussed and demonstrated within the chapter.

ACKNOWLEDGMENT

This work has been funded by the European Commission within the PEPERS project (IST-2004-026901). More information and the PDM support tool can be found at http://www.pepers.org

REFERENCES

Agarwal, D. A., Chevassut, O., Thompson, M. R., & Tsudik, G. (2001). An integrates solution for secure group communication in wide-area networks. In *Proceedings of 6th IEEE Symposium on Computers and Communications*, Hammamet, Tunisisa.

B'Far, R. (2004). *Mobile computing principles*. Cambridge Press.

BANKSEC (2000). EC funded project (IST-1999-20711). Retrieved on October 2, 2008, from http://www.atc.gr/banksec/

Berket, K., Essiari, A., & Muratas, A. (2004). *PKI*-Based Security for Peer-to-Peer Information Sharing. In *Proceedings of P2P 2004*, Zurich, Switzerland.

Bertolini, D., Busetta, P., Molani, A., Nori, M., & Perini, A. (2002). Designing peer-to-peer applications: An agent-oriented approach, In *Proceedings of the International Workshop on Agent Technology and Software Engineering*, Erfurt, Germany.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. Unpublished doctoral dissertation, University of California at Irvine, USA.

Kortuem, G. (2002). Proem: A middleware platform for mobile peer-to-peer computing. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R), 6*(4), 62-64.

Mouratidis, H., Giorgini, P., & Manson, G. (2003). An ontology for modelling security: The tropos approach. In *Proceedings of the 7th International Conference on Knowledge-based Intelligent Information and Engineering Systems*, Oxford, UK.

P2P ARCHITECT (2001). EC funded project (IST-2001-32708). Retrieved on October 2, 2008, from http://www.atc.gr/p2p_architect/index.htm

PEPERS (2006). EC funded project (IST-2004-026901). Retrieved on October 2, 2008, from http://www.pepers.org

Siveroni, I., Zisman, A., & Spanoudakis, G. (2008). Property specification and static verification of UML models. *Proceedings of the International Conference on Availability, Reliability and Security*, Barcelona, Spain.

Spanoudakis, G., Kloukinas, C., & Androutsopoulos, K. (2008). Dynamic verification and control of mobile peer-to-peer systems. In *Proceedings of the 3rd International Conference on Internet Monitoring and Protection (ICIMP)*, Bucharest, Romania, to appear.

Walkerdine, J., Melville, L., & Sommerville, I. (2002). Dependability properties of P2P architectures. In *Proceedings of the IEEE P2P*, Sweden.

Walkerdine, J,. Melville, L., & Sommerville, I. (2005). *Reference Architectures for Peer-to-Peer Applications* (Tech. Rep. COMP-009-2005). UK: Lancaster University, Computing Department.

Walkerdine, J., Lock, R., & Lock, S. (2006). D2 - analysis of security characteristics of peer-to-peer architectures. EC Project Deliverable, PEPERS IST-2004-026901.

Walkerdine, J., & Lock. S. (2007). Towards secure mobile P2P systems. In *Proceedings of the P2P Systems and Applications (P2PSA)*, Mauritius.

Section VI
Standards and Protocols