IoTility: Architectural Requirements for Enabling Health IoT Ecosystems

Wyatt Lindquist, Sumi Helal, *Fellow, IEEE*, Ahmed Khaled, and Wesley Hutchinson

Abstract— The increasing ubiquity of the Internet of Things (IoT) has the potential to drastically alter the way healthcare systems are utilized at home or in a care environment. Smart things offer new ways to assist in general patient wellness, such as promoting an active and healthy lifestyle and simplifying treatment management. We believe smart health things bring new requirements not typically addressed in traditional IoT systems, and that an architecture targeting these devices must address such requirements to fully utilize their potential and safe usage. We believe such an architecture will help improve adoption and efficacy, closing gaps between the variety of emerging health IoT systems. In this paper, we present a number of requirements we consider integral to the continued expansion of the digital health IoT ecosystem (Health IoT). We consider the current landscape of IoT in relation to these requirements and present solutions that address two pressing requirements: 1) democratizing mobile health apps (giving users control and ownership over their app and data), and 2) making mobile apps act and behave like any other thing in an IoT. We present an implementation and evaluation of these Health IoT requirements to show how health-specific solutions can drive and influence the design of more generalized IoT architectures.

Index Terms— Emerging technologies, Health, Requirements/Specifications, Ubiquitous computing

1 Introduction

In *The Importance of Being Thing Or the Trivial Role of Powering Serious IoT Scenarios* [1], we argued for the the importance of having explicit architectures for *things* in the Internet of Things, and pointed out that without first settling the quest for what a *thing* is or could be or do, we run the risk of presumptuous visions, or hypes, that can only fail the realities and limits of what is actually possible, leading to customer and consumer confusion as well as market hesitations. The article focused on the domain of "Personal" IoT and addressed key new requirements for *thing* architecture aiming at enabling their programmability into IoT applications. In this paper, we expand on our work in [1] and argue that for certain IoT application domains, additional, domain-specific requirements must be met and architected to enable IoT application development in that domain. We focus on the health application domain in which IoT is utilized, referred to as "Health IoT."

Let us first give a motivating example to explain why a thing architecture is needed. Imagine hosting a symposium for all the greatest minds in the world, with the ambitious task of curing diabetes. Large teams of people arrive, each from their own section of the world, each with their own area of interest and each with their own skillset. All are set to work, teams busily trying to progress their problem. But quickly, it is realised that the teams continue to work in their silos, unable to bridge the barrier of communication, unaware of the duplication of work and failing to benefit from the collective creation of the symposium. Any findings are passed through directly via set channels of communication. This is the current approach to the Internet of Things. We have many very smart things limited to their silos with the user unable to exploit the greater value of the whole. Now imagine the same scenario again, however, this time we ensure there are some essential requirements to maximise the productivity and achievements of the groups. For instance, we define a common scientific language for all attendees to display each team's interests, strengths, and studies. Team-defined mechanisms of interaction would then enable collaboration, sharing, and understanding among team members. In this environment, relationships develop, similar teams with analogous interests can discover mutually beneficial strength, or even work together on seemingly contradictory results seeking the truth to avoid scientific errors. Such relationships could lead to new ideas and outputs, where the symbiosis benefits the whole of the symposium.

Keeping the above example in mind, we argue that Health IoT things bring special and specific requirements not typically addressed in traditional IoT systems. We believe that any *thing* architecture targeting these devices must address such requirements to fully utilize their potentially collective and safe usage. We believe such an architecture will help improve adoption and efficacy, closing gaps between the variety of emerging health IoT systems in a highly fragmented and evolving market. Like the motivating example above, a successful architecture would enable the collective utility derived from the combined use of subsets of the *things* in the Health IoT.

- *Wyatt Lindquist is with the School of Computing and Communication, Lancaster University, Lancaster LA1 4WA, UK. E-mail: w.lindquist@lancaster.ac.uk*
- *Sumi Helal is with the School of Computing and Communication, Lancaster University, Lancaster LA1 4WA, UK. E-mail: s.helal@lancaster.ac.uk*
- *Ahmed Khaled is with the Computer Science department, Northeastern Illinois University, Chicago, IL 60625, USA. E-mail: aekhaled@neiu.edu*
- *Wesley Hutchinson is with the School of Computing and Communication and the School of Medicine, Lancaster University, Lancaster LA1 4WA, UK. E-mail: w.hutchinson@lancaster.ac.uk*

Otherwise, each *thing* will have its offering in isolation. We refer to this figure of merit in such architectures as *IoTility* or the ability to increase the collective utility of the Health IoT.

In section 3, we describe requirements for Health IoT that we broadly categorize into "device interactivity" and "user interactivity". In the former we address requirements that enable *thing*-to-*thing* interaction and autonomous inter-relations. In the latter, we address requirements for user-*thing* interactions that must be met to ensure proper, meaningful and safe use of the devices, the empowerment and enablement of users to easily use the devices, and the establishment of trust between the user and the Health IoT elements. In section 6, we present a high IoTility architecture for Health IoT emphasizing only parts that are most relevant to the requirements presented in section 3 through 5. This includes the IoT-Device Description Language (IoT-DDL), a machine- and human-readable language to provide the basis of cross communication and the establishment of inter-thing relationship. Section 7 presents implementation and evaluation of parts of the architecture. In this section, we present two unique frameworks/toolsets that meet the architectural requirements. The Runtime Development Environment (RIDE), allows end-users to dictate the high-level functionality of Health IoT applications they easily compose out of existing Health IoT devices; and the Mobile Apps As Things (MAAT) framework, allows developers of health mobile apps to utilize actionable keywords (AKWs) to enable the future IoTility of their mobile application, without the having to predict or plan for all potential future interactions. We conclude the paper in section 9.

2 Related Work

Managing an IoT ecosystem in a specialized environment requires a structured architecture fit for the job. Works such as NIST's *Network of Things (NoT)* [2] propose a foundational design for an IoT system. NoT defines a set of primitives describing the functionality of individual sensors and groups of devices, as well as how they may communicate. Laplante et al. [3] present another structured approach, specifically targeting IoT healthcare systems. The authors consider various use cases, such as managing dementia, and describe a set of privacy and safety requirements for a Health IoT system.

Catarinucci et al. [4] offer an architecture implementation, again targeting healthcare systems, that focuses on the interoperation of a variety of wireless protocols to collect and monitor patient data in a smart hospital scenario. The data can be accessed uniformly by healthcare providers, or monitored to send push notificaions to caregivers on critical sensor events. Our Atlas architecture [5] is another specialized IoT system, focusing on peronal IoT and the potential for interaction between devices. This architecture is described in further detail in section 6.

Facilitating interactions between IoT devices and *things* is a primary goal of many of these systems. The Social Internet of Things (SIoT) [6] describes group of smart object as a social network to mimic human behavior. Devices form social relationships over similar functionalities, vendors, or physical locations. If This Then That (IFTTT) [7] in the cloud and Things Talk to Each Other (TTEO) [8] on the device allow similarly allow users to compose services into rule based applications through a set of "if-then" triggers. These forms of interaction are expanded on in section 4.2.

Another goal of these IoT systems focuses on enabling interoperable usage of heterogenous devices. Initiatives such as the *Continua Design Guidelines* [9] provide a set of standards with an open implementation that manufacturers can follow in their health devices, creating a uniform base API across brands. In a similar fashion, the *Solid* [10] and *MyData* [11] services aim to improve health data accessibility, utilizing concepts such as decentralized storage and standardized formats. The importance of these features is discussed in section 4.1.

Within these goals, trust, privacy, and security also play a major role. NIST's NoT considers *thing* security at each level of their architecture, from sensors to user communication and triggers. Mahale et al. [12] present an access control system that calculates trust values based on parameters captured from a smart space. These trust values can then be used to manage user identity. Lomotey et al. [13] create a health information system to associate user identity with the various data streams gathered from sensors in a smart space. Section 5.3 furher details these issues.

3 Requirements for Health IoT

We identify a set of requirements we consider highly relevant to future *thing* architectures targeting Health IoT systems. We do not claim to provide a complete list of these requirements, but a selection of those we believe will allow such a *thing* architecture to maximize its IoTility in utilizing new digital health devices and the interactions and applications they enable. We group these requirements into two categories: 1) device interactivity, or how a device can expose its capabilities programmatically to application developers as well as cybernetically to other devices in a smart space, and 2) user interactivity, or how a device enables and guides an end user to properly (and safely) use it.

When considering the significance of device interactivity, one may reflect on the state of health data platforms such as Apple HealthKit [14]. This platform allows a user's phone to interact with supported devices, storing and displaying data in a unified interface. A user is provided with a level of assurance when buying a supported device—it will "just work" through its integration in the HealthKit app. However, this assurance hinges on this platform support: a device supporting only Google Fit or Samsung Health, for example, will be unable to interact with other HealthKit devices. These platforms offer users more control over their data and devices, but only in the context of their supported and closed ecosystem.

When considering the importance of user interactivity, one should look towards the plethora of personal health devices collecting precision data, which may generate erroneous or noisy data if used improperly or inaccurately—often a challenging task for the user. For instance, the Kardia [14] device can collect an electrocardiograph (EKG) sequence as the user places two fingers on each side of the device; proper finger placement is critical in receiving a quality reading. To facilitate this, the Kardia app exposes a familiar “signal strength”-style metric (figure 1), providing at-a-glance, feedback and guidance on finger placement to the user. Unfortunately, Kardia is an outlier in this regard; most devices lack the facilities to detect or react to inaccuracies and accept such interactions unconditionally.

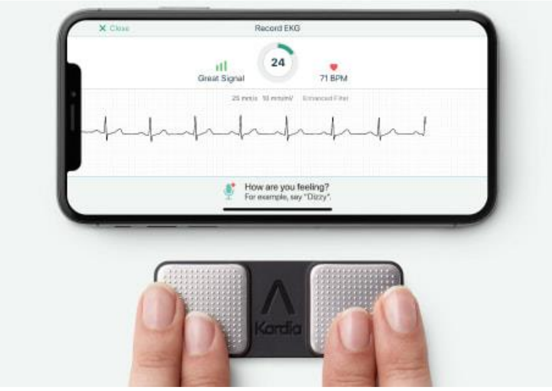


Fig. 1. The Kardia device and app interface.

To highlight both of these requirements, we consider the following near-future scenario throughout this section: following consultation with their physician, a patient’s risk of diabetes is highlighted as significantly raised. Suggestions are made for the patient to make lifestyle adjustments to reverse this risk. Just as medicine is prescribed, the physician also prescribes various IoT devices for the patient to obtain. The patient fulfills the prescription for a body-weight and body-fat sensing scale, a blood pressure monitoring device, and a glucose monitor. The patient chooses running as a means of exercise and so purchases smart soles to compliment the community of devices, as well as a pulse oximeter and a temperature monitor advertised as a more accurate measure of metabolic rate. Finally, the patient downloads a dieting mobile app to his smartphone. We use the elements of this scenario in the next sections to derive and explain our requirements for Health IoT.

4 Device Interactivity

A *thing* must have the ability to send information and receive commands before it can be useful in a digital health smart space. Many *things* have no physical user interface and limited potential for physical configuration; instead, they must utilize a more feature-complete parent device (such as an edge device or mobile phone) to act as the point of interaction with the user. A *thing* relies on this ability to interact with a parent device to fully represent its capabilities. To do this, the underlying thing architecture must provide not only hardware and software interfaces within the *thing*, but also the basis for communication with other devices. This architecture therefore becomes critical for the successful integration of that *thing* within a smart space.

To achieve this goal across the wide range of potential devices in health IoT, an architecture must be cognizant of how *things* may manifest themselves. For example, a *thing* may be a simple sensor, a higher-level device with a REST API, or even a full software system such as a mobile application. All of these *thing* types may perform the same functionality, such as reading a sensor value; however, the similarities stop there. Beyond utilizing different protocols, these *things* may perform their interactions in an entirely different way. The sensor *thing* may continuously emit its value as an electrical signal, the REST API device may perform its reading when an endpoint is invoked, and the mobile app may record a measurement based on the context of the phone's user's actions.

These different possibilities may be viewed as different "tiers" of interaction within the same system, where a *thing* architecture may always operate on, say, the REST API level of the interaction. However, with new devices constantly entering the IoT space, it is unreasonable to assume they will all operate in the same manner and expose the same capabilities. A health *thing* may not use a physical sensor or may be entirely represented within a mobile application. Instead, a *thing* architecture targeting digital health should consider the potential for inter-*thing* interaction across all of these forms.

4.1 Common Programming Interfaces (APIs)

Regardless of the context of its interactions, a *thing* must expose some form of API to allow it to communicate with the whole of the smart space. We believe the availability and utility of such an API is a key piece in determining how easily a new *thing* may be integrated into an existing smart space. Without an API, there is likely little to no way for a smart space to interact with said device. Rather, only the functionalities and system the device was explicitly programmed to utilize can be exploited.

Many devices in the current digital health landscape tend towards this pattern [14]. Lacking a true independent API, the device is tied to a specific ecosystem or subset thereof, creating "silos" of functionality segmented between manufacturers and vendors. While the potential for interaction inside the silo may be rich, such interactions cannot take place with devices outside that silo. If a user wants to fully utilize the potential of their smart space, they must stay within a silo of compatible devices. Such a situation is especially problematic when a specific device type does not exist within a user's chosen silo; the user is forced to use a device with potentially decreased functionality, or consider using a different silo.

Such an effect can be seen within the ecosystem of companion mobile apps for health IoT devices. A user with a collection of smart devices likely has a similar collection of apps on their mobile phone, with each device requiring its specialized app to make full use of its features. The natural progression of these silos is an ever-increasing number of apps on a user's phone as they acquire more devices: making it harder for the user to find the information they want, and making tasks such as showing multiple measurements from different devices at once more complicated.

Of course, eliminating these silos through a single standardized API is unlikely; vendors will always want to prioritize interactions between their own devices. However, even a limited subset of API compatibility, such as that defined by the Continua guidelines [9], could greatly improve a *thing's* ability to interact with its smart space. Consider the patient's temperature sensor from the scenario at the start of the section. Such a device is meant to be attached directly to the body, meaning it likely has little in terms of a physical interface (i.e., no integrated display to show the current temperature), and limited capacity for physical interaction (i.e., a sole button for toggling power).

The temperature readings and any needed configuration are instead exposed through a companion mobile app, which the user is required to download to use the sensor. Imagine, however, a user with accessibility concerns, where the companion app does not work properly on their specialized device. Without the app, the health device cannot be used properly, even though the hardware itself is functioning. However, in this case, the device exposes a minimal standard API with basic functionality. Perhaps the user holds their mobile device near the sensor, and receives usage instructions as well as the ability to perform a basic read from the sensor (thereby receiving their body temperature as desired). More advanced configuration features are not exposed through this API, but the user is still able to use the device, despite being unable to use the app.

Introducing more open APIs, however, does have implications regarding the safety and security of these smart devices [15]. Such concerns must also be carefully addressed; once a device provides data and receives commands more openly, the *thing* architecture must "pick up the slack" and ensure these APIs are not abused. Even when the API is being used properly, health devices must consider who is using the API; a primary user may be able to see readings from a device through its API, but these readings should not be available freely. These are other important requirements for the health IoT that are discussed in later sections.

We believe at least a minimal shared API is essential in mitigating the segmentation of *thing* devices. Ensuring functionality across a larger portion of users, in addition to providing users with more control over their data (such as with *Solid* [10] or *MyData* [11]) is essential when creating an effective thing architecture targeting digital health. While APIs give vendors less control over how their devices are used, they also have their own business cases: either making devices more desirable to consumers, or creating opportunities for platform services.

4.2 Relationships Between Things

Communication between *things* is a substantial part of an IoT ecosystem. The standardization discussed above is less impactful if considering only user-*thing* relationships. Once *things* are able to "speak" some level of a common language, they can interact not only with the user and edge, but also with each other. We believe this *thing*-to-*thing* interaction to be another critical part of a functional health IoT system. A *thing* with the potential to cooperate and utilize the capabilities of the smart space increases the capacity for meaningful interactions to occur.

As the number of devices in a smart space increases, explicitly programming them becomes more difficult; this is especially true in a health environment with a wide variety of medical sensors and devices. Even if all of the devices share a common interface, they still must be individually considered and programmed for. Beyond receiving data or sending controls directly, considering the potential for synergy between devices in a large smart space quickly becomes unreasonable. Allowing *things* to communicate between themselves has the potential to alleviate this burden.

Relationships, or logical links between functionalities offered by two or more *things*, allow for the creation of implicit interactions between smart space devices. A relationship may allow a thing to become a conditional element within a logic matrix, its output used as an input or to control another thing, as in IFTTT [7] and TTEO [8]. We believe that a *thing's* ability to form these types of relationships (especially when they can be formed as suggested by the *thing,* rather than explicit user intervention, such as the SIoT [6]), will allow users to focus on the high-level functionality of their smart space, leaving the low-level details to be taken care of by the architecture itself.

These kinds of relationships are especially useful in a health environment, where sensors may only record a part of a larger metric (such as how body temperature, blood pressure, etc. make up a patient's general vital signs), or may record the same measurement in conjunction with other sensors (such as reading pulse in different places on the body). Such grouping of information can be handled or specified between *things* through relationships, reducing the need for explicit programming and simplifying the use of the smart space at the edge, especially for personal health scenarios where the user may not want or know how to effectively manage their array of smart devices.

Using the initial scenario, consider the patient's blood pressure measuring device and pulse oximeter. Both of these devices are capable of recording, among their other measurements, the patient's pulse. In the traditional case, the user would be presented with two pulse readings, and would likely choose to use one over the other, possibly hiding or removing the second measurement. Imagine instead that these devices look to form relationships with other devices (whose specific form may not be known) that provide a pulse reading, allowing them to combine readings or keep each other in check. Such a situation improves usability and allows for a form of reliability across devices taking the same measurement (where services only need to be aware of the measurement itself, not the source device).

We believe relationships between *things* can play a large role in the effective programming and use of a health IoT system. When dealing with many devices in a smart space, allowing them to consider their position inside the smart space and the other functionalities being offered becomes valuable for both the end user and the operation of the smart space itself. An architecture handling these relationship-based behaviors can help supplement or simplify the creation of meaningful interactions in a smart space.

4.3 Thing-Like Mobile Applications

In much of the above discussion, the mobile app plays a prominent role in a smart space system. The number of these apps targeting digital health is significant: over 300,000 as of 2017 [16], [17], and still growing. In addition to self-contained health apps, many health IoT devices require a mobile device to host their companion app. However, despite their close interaction with *things* and the smart space, these mobile apps are not utilized fully: we believe they can be further integrated as *things* themselves. Considering mobile applications as "software *things*" (as opposed to hardware *things*, such as sensors, actuators, or other personal health devices) can open up new potential for meaningful interactions within a smart space.

In most situations, the mobile app either acts as a controller for other smart *things*, such as requesting data or sending commands, or manages information for the user, such as the aforementioned diet app. In terms of a smart *thing*, this is only half of the picture. Other *things* in a smart space cannot consider the app for any interactions beyond what it is programmed to do. Even if a health app's information could be used by another *thing* (especially one unrelated to the purpose of the app), the app most likely lacks a way to channel data to that *thing*. The app is missing a concrete API like those of other physical *things*, simply because most mobile apps are not designed to work this way.

For example, consider the patient's dieting app and their smart soles from the initial scenario. The user eats a meal, enters the information into their app, and receives an updated exercise plan. The user then must use another mobile app or other method for updating their smart soles' configuration to reflect their exercise parameters. Even though all the information is available implicitly within the smart space, the user must manually "transfer" parameters between the app and the *thing,* because the developer of the app did not consider this potential for interaction.

If the diet app had a *thing*-like API, the interaction could have progressed differently: after the user enters their information into the app, the smart soles see that the exercise plan has changed, updating their parameters automatically. In this case, the *thing* becomes the driver of the interaction, where it asks the mobile app for information through its *thing*-like API. Although the app was not developed with smart soles in mind, it was able to provide its information to the smart space and enable the interaction.

Compared to a normal hardware *thing*, a mobile app is likely much more capable in terms of features and the ability for the user to interact. *Things* in the smart space can potentially utilize a wide variety of sensors (for example, accelerometer and GPS) or engage the user through a touchscreen interface. This is especially interesting for lightweight hardware *things* with minimal physical interfaces, like the body temperature sensor mentioned above. Such functionality makes higher-level interactions available to *things* without the need for physical interfaces or a specially programmed app (thereby reducing the need for the "silos" of companion apps described in section 4.1.

We believe positioning mobile apps to behave more like *things* to be another critical requirement in an effective digital health *thing* architecture. As mobile applications become more prevalent in smart spaces, allowing them to behave like any other *thing* device will solidify them as an integral part of an IoT system. Such a feature creates new potential for meaningful interactions with the *things* in a smart space, allowing for *things* to easily interact and form relationships with mobile apps. *Thing*-like relationships between apps could allow users to "combine" app functionalities, or easily create new apps with the exact functionality they desire.

5 User Interactivity

Things in a smart space offer limited utility unless they can interact with and convey information to their users. While some health IoT systems may, for example, be set up professionally or pass their readings to the cloud for analysis before being viewed, many systems (especially in personal health situations) will see their data and functionality being accessed directly by users (coming from readings such as vitals and indicators of activity). In this case, interaction with the user is critical: aspects such as data acquisition are dependent on correct use of the device. In fact, it may be argued that proper user-*thing* interaction (with factors such as ease of use and required knowledge) constitutes a significant barrier hindering digital health adoption [18], [19].

In addition to using a device incorrectly, a *thing* architecture should consider the potential for lack of use as well. A user may interact with a device properly in terms of API and physical use, but not use it often enough, at the right times, or at all. In this case, a *thing* must have a concept of user motivation; that is, how the *thing* can interact to increase its chance of being used when needed. A *thing* may utilize markers such as a schedule or the behavior of other *things* in the smart space to hint at these opportunities.

To achieve this, an effective health IoT architecture should be able to understand and manage the potential for unreliability or error when dealing with input and output between a user and a *thing*. This includes validating input data and its proper acquisition, along with maximizing accessibility of the output data. We believe a health *thing* architecture needs to have the ability to monitor and correct the interaction process to ensure an interaction is completed properly and successfully: such an ability will allow the architecture to maximize its effectiveness across the spectrum of lay users in a health environment.

5.1 Input and Output Safety

Ensuring correct and reliable interactions with devices is a basic requirement of any IoT device. This requirement is especially critical in health IoT systems, where a sensor malfunction or other errors with the device could generate noisy or erroneous data—with potentially serious consequences. In addition to device reliability, proper acquisition of data is equally important in a digital health environment. For sensors, this means ensuring the user is using the device correctly (for example, such as proper placement of EKG electrodes). As mentioned previously, many sensor devices have limited ability to physically interact; they may not be able to signal proper use themselves.

Obviously, to monitor for proper use, a health *thing* must be architected to detect erroneous values or improper use symptoms in the first place. Depending on the specific sensor, this could be feedback in the form of binary information (such as valid or invalid), instantaneous feedback on accuracy (such as "signal strength" feedback), or others (such as instructions and corrections). If a thing can detect these parameters, it could provide feedback to the user, requesting that a reading be taken again or offering help on correct usage, rather than just recording these erroneous values as-is for the user or edge to handle later.

This user feedback becomes increasingly important in a health IoT ecosystem when considering the large number of health devices that may be in use. As a lay user acquires more devices, it becomes harder to stay familiar with all of the devices and their proper use instructions. Consider how the patient was prescribed health *things* in the initial scenario. A user setting up a device they did not choose personally may be less likely to initially understand its proper use. While they could read an instruction manual for the device, offering feedback directly through their Health IoT device would likely increase the chances of the user seeing the information and using the device properly.

Like sensors, actuator things depend on proper use by the user. While (simple) actuators do not transmit data, they receive commands that, when used improperly, could physically damage components or expose the user to potential harm. For example, a hybrid closed-loop insulin pump provides the user with an insulin dose dependent on the reading from their continuous glucose monitoring device [20], [21]. Improper use or placement of the monitor or pump could result in an incorrect dose being delivered. Preventing these actuations from occurring is another important requirement in a digital health smart space; as mentioned above, when a user is unfamiliar with a device, the potential for error or malfunction must be limited.

Actuation commands may be validated through *constraints*, or limitations on the frequency, magnitude, etc. of an API invocation. These run-time enforcements could be provided by developers, vendors, or owners to help govern safer things interactions that can be fine-tuned for specific smart space deployments. Such enforcements would help prevent issues (at least those that could be caused by the user) before the device fails—in the case of the hybrid insulin pump, hardware constraints have resulted in an impressive safety profile in testing [21], [22]. Additionally, in the case that a device does fail, constraints can also be capable of defining a "fail-safe" mode [23], where the final resting state of a device has minimal potential for harm.

The hybrid insulin pump is just one example of how automated devices can be safe despite controlling high-risk activities. More closed-loop monitoring and dosing devices are beginning to emerge, such as activity sensors providing input for levodopa dosing in Parkinson's disease. The use of objective sensor data to dictate medication dosing must be undertaken with careful consideration, but has the potential to provide personalized treatment regimens with better safety outcomes [24].

Current work focuses mainly on data validation and monitoring of sensor data in health scenarios. O'Donoghue et al. [25] present a Data Management System that focuses on data validation and consistency of different IoT sensors, as well as on how to choose which information is relevant to the user. Yang et al. [26] focus on data validity and reliability in a set of wearable health devices and mobile apps.

We believe an architecture should carefully consider how data and commands enter and leave a smart device. Providing feedback on the quality of produced data and validating inputs for safety and correctness are two facets that are likely to be valuable components of a health IoT smart space. These would allow *things* to handle problematic interactions before they are utilized by the target device or other devices in the smart space, strengthening assurance to the user and trust in the device. Integrating these can help simplify user interactions and allow the user to better understand their smart space.

5.2 Notifications and Reminders

When considering the relationships between user interactions and validating device input/output, another requirement becomes prominent: notifications. The previous section discusses providing feedback and assistance to users based on their use of a device. Often, the user may not be viewing the output of a sensor real time; therefore, during events such as abnormal or erroneous readings, the device may need to interact with the user another way. Depending on a device's ability to convey information, it may be more efficient to broadcast a notification where a more capable *thing*, such as one with a screen, sound, or other output could display the information. This is especially relevant when considering the potential of mobile apps as *things*.

As mentioned above, a critical part of a sensor monitoring its data quality or an actuator limiting its input is displaying this information to the user. That information can then be used to adjust how the device is being utilized, improving these interactions with the device. A concept of notifications would allow this information to be viewed by the user in a uniform way. In situations like these where this error data is ephemeral, notifications allow the user to review the information after the fact, without having to store it permanently. Notifications could also be extended outside of a local smart space, to provide family or care practitioners info on critical events.

A specific form of notifications, reminders, is especially important for user empowerment in a health IoT smart space. In addition to conveying its data, a device may want to assist the user in performing actions at certain points in time. For a health device, this may include notifications for proper use, maintainance, calibration, etc. These events are time sensitive, and likely to be known by the device before hand. By sending a notification at the appropriate time, a device does not need to rely on the user remembering a schedule and can increase the likelihood of proper use.

For example, consider the glucose monitor from the initial scenario. To take an optimal reading, the user must measure their blood sugar about 20 minutes after eating. The burden of remembering to measure at the appropriate time, even though meal information is likely available within the smart space, through the dieting app. A *thing* architecture with notification and reminder support could enable the dieting app to schedule a reminder for the glucose monitor after the user inputs a meal. After 20 minutes, the reminder would appear on the user's smartphone, increasing the likelihood of proper use.

In addition to interacting with the user, notifications can also be used between devices to help signal and trigger more complex interactions. A sensor that normally only sends out the values it records may also be able to send information about its state, such as information about itself and its functioning or when the user changes how it is being used. These notifications could be picked up by other devices that rely on the user performing a specific action before they should begin their own interaction. These events, along with the inter-*thing* relationships mentioned above, can provide the groundwork for enabling meaningful interactions within a smart space.

Phone-based notifications are a popular component in architectures targeting health today. Catarinucci et al. [4] present a system for a smart hospital that uses push notifications to send sensor events to caregivers. Tcarenko et al. [27] introduce a system to send notications remotely to caregivers on critical health events, such as a fall. Some systems integrate mobile-like notifications further; Kubitza et al. [28] describe an architecture that allows notifications to be displayed contextually through capable smart devices, reducing reliance on the mobile phone.

We believe an architecture should include some form of notifications or message passing between *things* in addition to the normal means of sending commands or receiving data. The above sections discuss allowing things to react to smart space events even when the source or target device is not explicitly known or programmed for. Notifications are a convenient way to achieve this, in addition to providing an additional way for users to view important information about their smart space. Building off of the idea of mobile apps as *things*, notifications are already a familiar concept to users that could be extended to work between *things* and mobile devices.

5.3 Managing Identity

Many health IoT *things* are designed with a single user in mind (for example, wearable sensors). However, some devices (such as larger, non-wearable ones) may be designed to be used concurrently by multiple users. This is especially true in personal health scenarios, where in a single smart space, a device may be used by a group, such as a family or entire household. In these cases, the thing must have some notion of who it is being used by, so that the data it produces or the functionality it performs can be associated with the correct user within the smart space. Even *things* made for single users may utilize some form of identity, especially in relation to health IoT. In a multi-user smart space, a specific smart *thing* may be concerned with privacy: only a specific user should have access to the device's data or functionality.

This understanding of identity is another important component of a *thing* architecture targeting health IoT. A sensor device that changes users would likely see a different range of normal values (for example, blood pressure), which could create issues if the device is unaware the readings are now from a different user. Even if the smart space edge or cloud is capable of handling multiple users, a shared device must also be aware of this logic: at a minimum, the device must be able to inform the smart space about the current user identity.

Building upon the initial scenario, consider when the user receives all their prescribed devices. Upon activating each device, it is assigned with some form of secure user-specific identifier. During use, the devices may only communicate with other devices using the same user identifier, preventing unauthorized access. This concept of identity is necessary for the user, who lives with a roommate in a shared dwelling. Even though both have smart *things* on the same network, the roommate is unable to access the APIs of the user's *things* because the architecture controls access based on the user identifier.

Identity management for a health *thing* consists of two main stages: provisioning and use. During provisioning, or setup of the device, a user identity must be assigned or created. This identity could potentially be created with little user intervention, sourcing parameters from the user or the smart space [12]. For single-user devices, this involves associating a specific user with the device and its data, while for multi-user devices, this might involve the creation of separate data profiles for each user. For example, a simple temperature sensor may be configured to send its data to a user-specific endpoint. The other stage, use, involves identifying who the current user is, whether they are authorized, and how any generated data or received commands should be handled, such as in the system by Lomotey et al. [13]. For example, a smart watch may require a password or nearby unlocked phone before it can be used.

We believe an architecture targeting digital health should be able to understand and manage user identity. User privacy and information security are critical parts of a health IoT system, both of which have a basis in the appropriate handling of user identity. Considering how devices are initially set up and configured, and how they could be used by members of a smart space is another important role of a digital health IoT architecture.

6 High IoTility Architecture for Health IoT

In response to the above, we present our Atlas Thing

Architecture, which includes functionality we believe begins to satisfy some of these new requirements. The Atlas architecture consists of several works focusing on different issues in IoT; however, we will focus only on the subset that relates directly to the above requirements. This includes the architecture itself, focusing on communication between *things*, the IoT Device Description Language (IoT-DDL), handling dynamic *thing* APIs, and the Inter-Thing Relationships framework, defining a concrete set of relationships between *things*.

6.1 IoT-Device Description Language (IoT-DDL)

In the absence of a device description language, significant effort is required to interact and manage the wide heterogeneity of *things* that can exist in a smart space. At the same time, the *thing* description should be part of the *thing* itself to facilitate a *thing's* smooth migration between smart spaces and to enable *thing*-to-*thing* ad-hoc interactions. The IoT-DDL [29] is a machine- and human-readable XML-based descriptive language used to describe, through a set of attributes, values, and parameters, a *thing* in a smart space. The IoT-DDL describes the *thing* in terms of the *thing's* identification information, resources, inner entities (for example, sensors, actuators, and software-based functionalities), along with the services such a *thing* offers to other smart space users and devices. Each *thing* entity provides a subset of these services through a set of well-defined interfaces (APIs). Such configuration scheme is then uploaded to the *thing* to enable it to self-discover its capabilities and engage with the surrounding IoT ecosystem.

A *thing* can also use the IoT-DDL to describe how it is socially related and linked to other *things*. Using identification attributes such as model, vendor, etc., the *thing* can describe how offered services can be logically and functionally tied. Such a social network of logically connected *things* can help guide the creation of new meaningful interactions. The IoT-DDL also enables explicit description of such logical social bonds and functional relationships.

The IoT-DDL uses this human- and machine-readable format to provide the basis for a uniform API across *thing* devices. It limits the creation of silos between groups of devices (as mentioned in the above requirements), by enabling *thing* interfaces to be defined and modified by the vendor, developers, and end users. This helps ensure a greater compatibility among smart *things*, even if they were not originally programmed for each other explicitly.

6.2 Atlas Thing Architecture

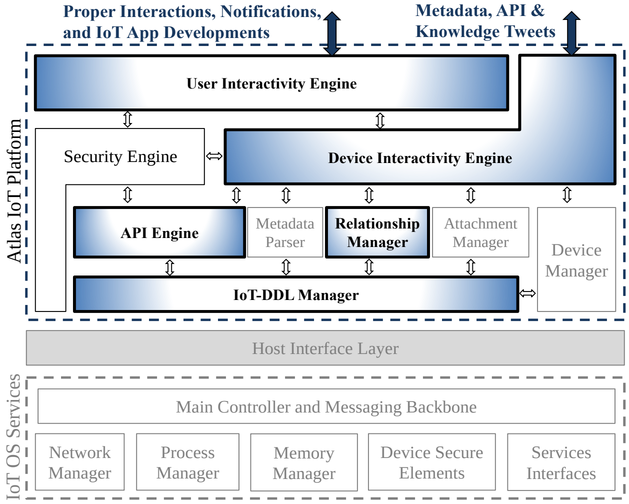


Fig. 2. The Atlas Thing Architecture.

The current IoT platforms and architectures link the access of *things* to a central point (for example, cloud platforms or the edge) where direct communication between things is hardly supported. These vendor-constricted connections narrow down the opportunity to integrate *things* from different vendors seamlessly into the smart space. This restricted paradigm ignores the potential for devices to communicate with each other, in addition to cloud platforms and the edge. The Atlas Thing Architecture [5] is a set of software operating layers and modules that utilizes the capabilities of the IoT-DDL (discussed in the previous section), mounted onto a thing to provide new functionalities it requires to engage and interact with other things, platforms, and IoT scenarios.

The architecture, illustrated in figure 2, consists of three main layers: Atlas IoT platform, host interface layer, and IoT OS services. The IoT OS services are the basic services provided by the *thing's* OS (e.g. process execution and management, network modules, memory units, and I/O interfaces). The Atlas IoT platform represents the logical layer of the architecture and provides new IoT services not currently provided by the *thing's* OS. Such new services focus on the descriptive and semantic aspects of *things* to better enable engagement, interaction, and programmability in an IoT. Such services enable a *thing* to: 1) self-discover its characteristics, resources, and capabilities through the uploaded IoT-DDL, 2) dynamically generate outward services and formulate their appropriate APIs based on information in the IoT-DDL, and 3) enable secure interactions between *things* and users in new IoT applications and scenarios, including those in which smart *things* speak different "languages," using a protocol translator attachment [30]. The host interface layer, the middle layer of the architecture, shields the platform and provides the portability and interoperability features needed. This layer manages the internal interactions between the Atlas IoT platform and the set of services provided by the underlying OS.

This communication between devices exists as a set of information- and action-based interactions. Information-based interactions (referred to as *tweets*) enable a thing to announce its metadata, API, and knowledge of the smart space to nearby *things*. A *thing* can use these *tweets* to describe what it is, what it does, and what it knows. Action-based interactions are utilized to execute relationships and provide notification-like messages to *things* and users.

The Atlas Thing Architecture focuses on enabling *device interactions*; by utilizing different communication standards and a uniform API interface, the chances for meaningful inter-*thing* interactions are increased. The architecture can run on a range of devices, from sensor boards to Linux-based systems to Android devices, enabling intercations between *things* with varying capabilities.

6.3 Inter-Thing Relationships Framework

The Inter-Thing Relationships Programming Framework [31] utilizes both the Atlas Thing Architecture and the IoT-DDL to build a distributed programming ecosystem for the social IoT. The framework broadens the social bonds (*thing*-level relationships) between *things* according to their identification attributes (for example, vendor or *things* collocated in the same space) and utilizes a new set of relationships between the offered services (for example, a competitive relationship or a relationship that extends functionalities) that we believe can empower developers to program a much wider class of meaningful applications.

A *thing* can also use the IoT-DDL to describe how it is socially related and linked to other *things*. Using identification attributes such as model, vendor, etc., the *thing* can describe how offered services can be logically and functionally tied. Such a social network of logically connected *things* can help guide the creation of new meaningful interactions. The IoT-DDL also enables explicit description of such logical social bonds and functional relationships.

The framework introduces services (abstractions of the functions offered by a *thing*), relationships (abstractions of how services are linked together), and recipes (abstractions of how services and relationships build up an interaction) as the primitives for an Atlas IoT application. The framework also defines *filter, match,* and *evaluate* as three operators that functionally define how the primitives are wired. The description of an IoT application within the framework utilizes a set of semantic rules that evaluate the correctness of the developer's established application.

These relationships within the framework can be utilized by vendors in the IoT-DDL, defined by developers while building IoT apps, or dynamically inferred from the exchanged knowledge (*tweets*) between *things.* This ability to discover and infer new links between *thing* services allows more meaningful interactions to develop with less intervention needed from the user.

7 Implementation and Evaluation

In addition to the core Atlas architecture components, we also present two extensions to the core architecture that have been developed to satisfy a specific requirement mentioned within the previous sections. The first extension, the Runtime Development Environment (RIDE), focuses on utilizing relationships between things, allowing users to develop new smartphone apps composing their *things*. The second, Mobile Apps As Things (MAAT), focuses on providing smartphone apps with *thing*-like capabilities and making them available to the smart space. A brief description of each is provided, along with some preliminary evaluations and results.

7.1 Runtime Development Environment (RIDE)

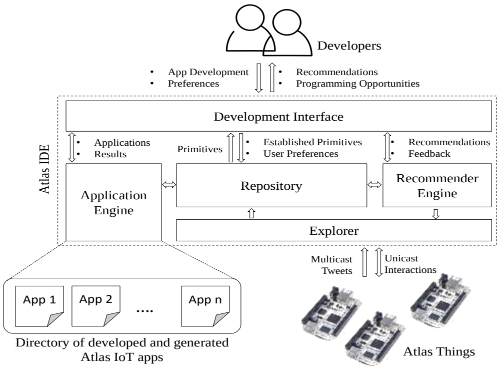


Fig. 3. High-level architecture for Atlas RIDE.

RIDE is a development environment, runtime system, and interactive tool for end users to develop and build IoT apps. It extends the Inter-Thing Relationships framework described in section 6.3, utilizing the Atlas architecture and IoT-DDL to build a distributed programming ecosystem that utilizes a set of concrete relationships for the development of a wider class of domain-related IoT apps.

Using RIDE, a developer can: 1) continuously listen to the *things* in the smart space, visualizing available services and relationships; 2) establish new relationships and applications; 3) infer new opportunities from existing services and relationships; and 4) set preferences for functionalities and services to guide the inferences of these new opportunities. RIDE also accepts a description for a new application and generates an independent Android mobile app that communicates with the smart space. The IDE, as illustrated in figure 3, targets smartphone users with no programming experience to easily create new smart space IoT apps with a touchscreen interface.

The developer uses the Development Interface, utilizes primitives (from the Inter-Thing Relationships framework) from the Repository to establish new IoT applications, while the Inference Engine discovers new relationships, recipes, and programming opportunities from existing primitives. The Inference Engine also holds developer preferences based on feedback from previously inferred applications, which guides future inference. The Application Engine checks the validity and correctness of a created application (either established manually or inferred by the IDE) before generating an Android app executable through an external on-cloud service, based on an XML description of the chosen primitives. Each generated application, shown in figure 4, is governed by a set of semantic rules.

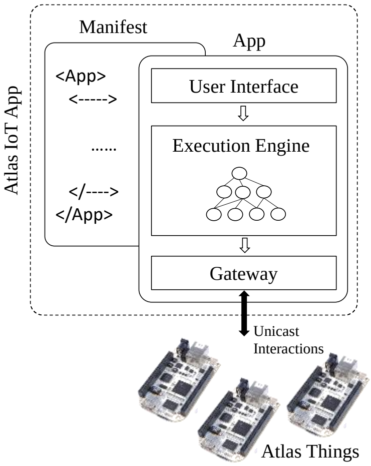


Fig. 4. The structure of a generated IoT mobile app.

The Atlas RIDE prototype is built for Android-based smartphones, where generated IoT applications are fully independent Android apps. We performed a set of experiments using a Nexus 9 Android device to evaluate and benchmark the feasibility of the proposed IDE. The energy consumption of RIDE under various conditions was measured and compared against that of the background OS processes and tablet hardware. The difference in these values was used to determine the energy consumption of the IDE.

One such measurement is shown in figure 5. This measurement shows the runtime energy consumption of a generated application—that is, the power required to communicate back and forth with the APIs of the other Atlas smart space things offering the required services. Parameters are sent to a thing and a response value is received before repeating this interaction with the next endpoint, until all services and relationships in the recipe have been executed. As the number of services increases, the energy consumption increases as well, but remains negligible overall.

In our *DIY Health IoT Apps* demo [34], we utilized RIDE in a health IoT scenario. The scenario simulated a health smart space with a temperature sensor *thing*, a pulse oximeter *thing*, a bodyweight scale *thing*, and a fitness mobile app with *thing* capabilities. Atlas RIDE was used to generate two applications: 1) an app that displayed combined readings from the temperature sensor and pulse oximeter; and 2) an app that automatically passes the reading from the bodyweight scale into the fitness app for calculations. None of the *things* were pre-configured for these interactions; they are both handled through the Inter-Thing Relationships framework.

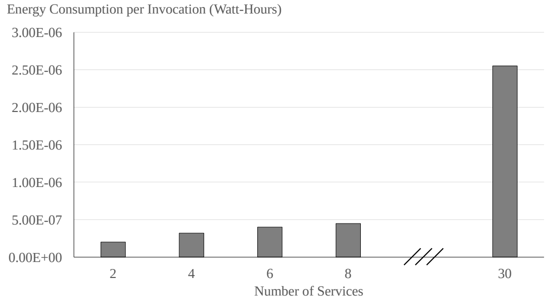
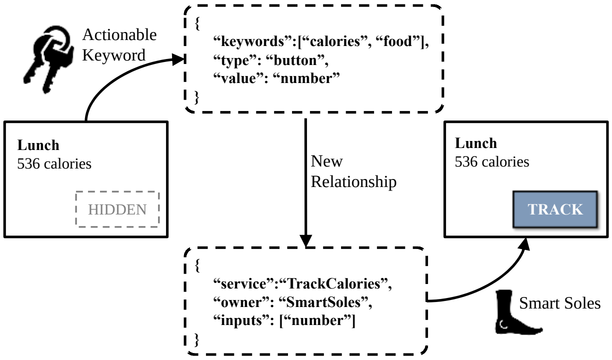


Fig. 5. Energy consumption of generated personal IoT applications.

7.2 Mobile Apps As Things (MAAT)

The Atlas architecture as described in section 6 has mainly focused on enabling hardware devices to be *things*. To complement this, we introduced MAAT, a framework that allows mobile apps to behave as traditional *things* and seamlessly communicate with existing hardware *things* in a smart space. While the framework does not provide an app with the full feature set of the hardware Atlas platform, it achieves parity with core features such as API-ing and inter-thing interaction. The framework also considers the role of the mobile developer, who may not be familiar with IoT or want to waste time adding complex IoT support. To this end, MAAT also introduces a programmable description called an *Actionable Keyword (AKW),* along with an IDE plugin to minimize changes to a developer's workflow.



```
{
  "keywords":["calories", "food"],
  "type": "button",
  "value": "number"
}
```

```
{
  "service":"TrackCalories",
  "owner": "SmartSoles",
  "inputs": ["number"]
}
```

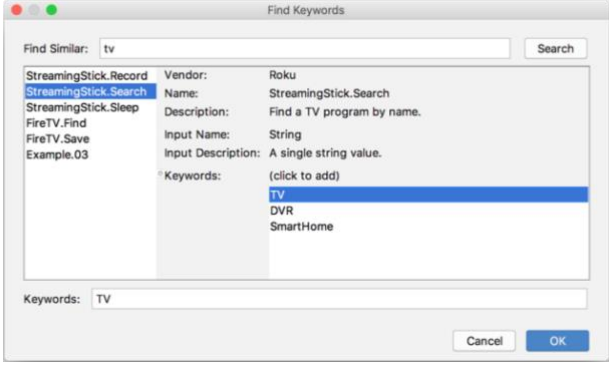
Fig. 6. The Actionable Keyword lifecycle.

Receiving capabilities from a *thing* in the form of API declarations, as was utilized in our previous apps-as-things demo [33], provides the app with the information it needs, but places a large burden on the mobile developer. The developer must know the exact API to integrate with before hand, and must anticipate how the *thing* will manifest within the UI and behavior of the app. If these parameters are not known, the context of the interaction likely must be handled "on top" of the existing UI, such as in a pop-up or entirely new interface.

Consider, for example, the scenario from section 3. The smart soles have a function to track how much of a meal the patient has burned off by exercising. To calculate this, the soles require the total calorie count of the meal. The dieting app can provide this value; however, the developer did not consider the potential for interaction with smart soles. Even if the app can give the soles calorie information based on its API, it still lacks the context of when and where this value might come from.

Instead, MAAT allows the developer to specify potential data from their app to be used in a smart space. In this situation, the developer knows the calorie information from the user's recent meals could be useful, but does not have a target device in mind. MAAT allows the developer to say, "the user is interested in this calorie count," rather than wait for a device to announce, "I can do calculations with calories." By specifying the data, the developer announces that each listed meal is potential input for a *thing*.

This data and context information is represented within an *actionable keyword.* A single piece of data (one of the recent meals) is associated with a user interface element (a button) to trigger a future *thing* service. This relationship is represented in figure 6; the developer configures an *AKW* containing calorie information from a single meal. The trigger button remains hidden until the data is associated with a *thing* service. Once the smart soles discover this opportunity and offers its tracking service, the button appears (with a label specified by the soles), which will send the specific calorie value to the sole's service when tapped.



Find Similar: tv

StreamingStick.Record
StreamingStick.Search
StreamingStick.Sleep
FireTV.Find
FireTV.Save
Example.03

Vendor: Roku
Name: StreamingStick.Search
Description: Find a TV program by name.
Input Name: String
Input Description: A single string value.
Keywords: (click to add)
TV
DVR
SmartHome

Keywords: TV

Fig. 7. Keyword search within the IDE plugin.

Finding the association between an *AKW* and a potential *thing* service, however, is difficult. To solve this, along with the input data, and *AKW* also specifies a set of keywords that are semantically compared by the *thing*. For example, the smart soles might look for numeric input with "calorie" and "food" keywords. These keywords build off of the descriptive keywords from Atlas; MAAT also includes an IDE plugin to search a repository of keywords and input data scraped from a database of IoT-DDL specifications. This interface is shown in figure 7.

Due to their direct interaction with the mobile app's UI, *actionable keywords* must be able to be processed quickly. Any delays could be confusing or cause the app to appear sluggish; potential relationships should appear smoothly as the user navigates throughout the app. In figure 8, we analyze the total time between broadcast and formation of a relationship, for varying numbers of *AKWs*. Even with a very large number of *AKWs*, the total response time remains reasonable at about half a second.

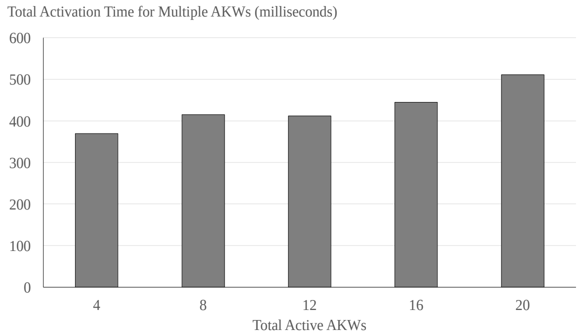


Fig. 8. Total response time of multiple active AKWs.

8 Discussion and Future Work

In this section, we discuss several issues of scope, limitations, and future work. First, the presentation of the requirements introduced in this paper is deliberately focused on the ideas behind them, the reasons they are needed, and the motivations of their potential impact. However, they can be further formalized and tested using a requirement engineering process [34], which is outside the scope of this paper. Formalizing the requirements will facilitate communication with personal health devices standards such as the IEEE 11073 and Continua Alliance [9] in the hope that such requirements may be adopted and included within the standard bases. Engaging standards organizations in our work will ensure practical pathways to widespread adoption, and more importantly, tests and certifications that these requirements are met; processes which are often within the remit of these organizations.

Second, while we engage general practitioners as a key stakeholder in arriving at the user interactions requirements in this paper, additional stakeholders, including other health professionals and the end users themselves, can further refine our requirements or add to them. We are currently conducting a large-scale study on user interaction with digital health involving a multitude of commercially available devices and a sizable number of users and health professionals. We are hopeful this work in progress will capture more broadly any elements we may have missed in our work on user interaction requirements so far.

Third, to best focus on the new requirements (especially in regard to user interactions), we limit the scope of the paper to users without special needs. However, additional accessibility requirements and special interface design for individuals with special needs remain important and should be further addressed. While such requirements are not discussed directly, considering they are their own area of specialization and outside the scope of this paper, we completely acknowledge their importance and the need to further develop them as Health IoT progresses into the future.

9 Conclusions

Health IoT *things* bring new requirements not typically addressed in traditional IoT systems. We presented numerous examples to demonstrate this argument, along with a detailed analysis of new requirements, which we classified into device interaction requirements and user interaction requirements. The former is needed to enable inter-device interaction, communication, and most importantly inter-relationships. It is also needed to enable mobile apps to be and act as other health IoT *things*. This is important given the large number of health mobile apps. We also analyzed user interaction requirements showing how the device could support and empower the user to use the device properly and safely, and how users could gain control over their mobile apps and devices. We presented an architecture targeting Health IoT devices that address the analyzed requirements to fully utilize their collective and safe usage. We considered the current landscape of IoT in relation to these requirements and presented solutions that address two pressing requirements: 1) democratizing mobile health apps (giving users control and ownership over their app and data), and 2) making mobile apps act and behave like any other *thing* in an IoT. We presented an implementation and evaluation of these Health IoT requirements to show how health-specific solutions can drive and influence the design of more generalized IoT architectures.

REFERENCES

[1] S. Helal, A. Khaled and W. Lindquist, "The Importance of Being Thing Or the Trivial Role of Powering Serious IoT Scenarios," in *Proceedings of the IEEE ICDCS Conference 2019*, 2019.

[2] J. Voas, "Networks of 'Things'," NIST Special Publication, 2016.

[3] P. Laplante, M. Kassab, N. Laplante and J. Voas, "Building Caring Healthcare Systems in the Internet of Things," *IEEE Systems Journal,* vol. 12, no. 3, pp. 3030-3037, 2018.

[4] L. Catarinucci, D. De Donno, L. Mainetti, L. Palano, L. Patrono, M. Stefanizzi and L. Tarricone, "An IoT-Aware Architecture for Smart Healthcare Systems," *IEEE Internet of Things Journal,* vol. 2, no. 6, pp. 515-526, 2015.

[5] J. King, R. Bose, H.-I. Yang, S. Pickles and A. Helal, "Atlas: A service-oriented sensor platform: Hardware and middleware to enable programmable pervasive spaces," in *31st IEEE Conference on Local Computer Networks*, 2014.

[6] L. Atzori, A. Iera, G. Morabito and M. Nitti, "The Social Internet of Things (SIoT) - When social networks meet the Internet of Things: Concept, architecture and network characterization," *Computer Networks,* vol. 56, no. 16, 2012.

[7] "IFTTT", Available: ifttt.com

[8] J. Yun, I.-Y. Ahn, S.-C. Choi and J. Kim, "TTEO (Things Talk to Each Other): Programming smart spaces based on IoT systems," *Sensors '16,* vol. 4, p. 467, 2016.

[9] Personal Connected Health Alliance, "Continua Design Guidelines," Available: www.pchalliance.org

[10] MIT, "Solid," 2017. Available: solid.mit.edu.

[11] "MyData," 2018. Available: mydata.org/mydata-101.

[12] P. Mahalle, P. Thakre, N. Prasad and R. Prasad, "A Fuzzy Approach to Trust Based Access Control in Internet of Things," in *3rd Int'l Conf. on Wireless Comm., Vehicular Technology, Information Theory and Aerospace and Electronic Systems*, 2013.

[13] R. Lomotey, J. Pry and S. Sriramoju, "Wearable IoT data stream tracability in a distributed heath information system," *Pervasive and Mobile Computing,* vol. 40, pp. 692-707, 2017.

[14] "Apple HealthKit". Available: developer.apple.com/healthkit.

[15] "AliveCor Kardia". Available: www.alivecor.com.

[16] D. Dimitrov, "Medical Internet of Things and Big Data in Healthcare," *Healthcare Inform. Res.,* vol. 22, no. 3, pp. 156-163.

[17] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant and K. Mankodiya, "Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare," *Future Generation Computer Systems,* vol. 78, no. 2, pp. 659-676, 2018.

[18] IQVIA, "The Growing Value of Digital Health," 7 November 2017. Avail.: www.iqvia.com/institute/reports/the-growing-value-of-digital-health.

[19] Research2Guidance, "mHealth App Economics 2017," Avail.: research2guidance.com/product/mhealth-economics-2017-current-status-and-future-trends-in-mobile-health.

[20] M. Mackert, A. Mabry-Flynn, S. Champlin, E. Donovan and K. Pounders, "Health Literacy and Health Information Technology Adoption: The Potential for a New Digital Divide," *Journal of Medical Internet Research,* vol. 18, no. 10, 2016.

[21] M.-P. Gagnon, P. Ngangue, J. Payne-Gagnon and M. Desmartis, "m-Health adoption by healthcare professionals: a systematic review," *Journal of the American Medical Informatics Association,* vol. 23, no. 1, pp. 212-220, 2016.

[22] S. Weinzimer, G. Steil, K. Swan, et al., "Fully Automated Closed-Loop Insulin Deliver Versus Semiautomated Hybrid Control in Pediatric Patients with Type 1 Diabetes Using an Artificial Pancreas," *Diabetes Care,* vol. 31, no. 5, pp. 934-939, 2008.

[23] R. Bergenstal, S. Garg, S. Weinzimer, et al., "Safety of a Hybrid Closed-Loop Insulin Delivery System in Patients with Type 1 Diabetes," *JAMA,* vol. 316, no. 13, pp. 1407-1408, 2016.

[24] R. Bergenstal, D. Klonoff, S. Garg, et al., "Threshold-Based Insulin-Pump Interruption for Reduction of Hypoglycemia," *N. Engl. J. Med.,* vol. 369, no. 3, pp. 224-232, 2013.

[25] W. Dunn, "Designing safety-critical computer systems," *Computer,* vol. 36, no. 11, pp. 40-46, 2003.

[26] I. Thomas, M. Alam, F. Bergquist and e. al., "Sensor-based algorithmic dosing suggestions for oral administration of levodopa/carbidopa microtablets for Parkinson's disease: a first experience.," *J Neurol.,* vol. 266, no. 3, pp. 651-658, 2019.

[27] J. O'Donoghue and J. Herbert, "Data Management within mHealth Environments: Patient Sensors, Mobile Devices, and Databases," *Journal of Data and Information Quality,* vol. 4, no. 1.

[28] P. Yang, D. Stankevicius, V. Marozas, Z. Deng, E. Liu, A. Lukosevicius, F. Dong, D. Xu and G. Min, "Lifelogging Data Validation Model for Internet of Things Enabled Personalized Healthcare," *IEEE Transactions of Systems, Man, and Cybernetics: Systems,* vol. 48, no. 1, pp. 50-64, 2016.

[29] I. Tcarenko, T. Gia, A. Rahmani, T. Westerlund, P. Lijeberg and H. Tenhunen, "Energy-Efficient IoT-Enabled Fall Detection System with Messenger-Based Notification," in *Wireless Mobile Communication and Healthcare*, 2016.

[30] T. Kubitza, A. Voit, D. Weber and A. Schmidt, "An IoT Infrastructure for ubiquitous notifications in intelligent living

environments," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016.

[31] A. Khaled, A. Helal, W. Lindquist and C. Lee, "IoT-DDL - Device Description Language for the "T" in IoT," *IEEE Access,* 2018.

[32] A. Khaled and S. Helal, "Interoperable Communication Framework for Bridging RESTful and Topic-based Communication in IoT," *The Future Generation Computer Systems Journal Special Issue on "Internet of Things: Communications, collaborations and services in networks of embedded devices,* 2018.

[33] A. Khaled, W. Lindquist and A. Helal, "Service-Relationship Programming Framework for the Social IoT," *Open Journal of Internet of Things (OJIOT),* vol. 4, no. 1, pp. 35-53, 2018.

[34] A. Khaled, W. Lindquist and S. Helal, "DIY Health IoT Apps," in *16th ACM Conf. on Embedded Networked Sensor Systems*, 2018.

[35] S. Helal, A. Khaled and V. Gutta, "Atlas Thing Architecture - Enabling Mobile Apps as Things in the IoT," in *23rd ACM Int'l Conference on Mobile Computing and Networking*, 2017.

[36] G. Kotonya and I. Sommerville, Requirements Engineering: Processes and Techniques, Wiley Publishing, 1998.

[37] S.-H. Chang, R.-D. Chiang, S.-J. Wu and W. Chang, "A Context-Aware, Interactive M-Health System for Diabetics," *IT Professional,* vol. 18, no. 3, pp. 14-22, 2016.

[38] P. Mahalle, B. Anggorojati, N. Prasad and R. Prasad, "Identity Authentication and Capability Based Access Control (IACAC) for the Internet of Things," *Journal of Cyber Security and Mobility,* vol. 1, no. 4, pp. 309-348, 2013.

Wyatt Lindquist received the B.Sc. degree in computer eng. from Univ. of Florida, USA, in 2017. He is currently pursuing the Ph.D. degree in computer science at the School of Computing and Comm., Lancaster Univ., UK. His current research interests include IoT, operating systems, and embedded systems with applications in digital health.

Abdelsalam (Sumi) Helal received the Ph.D. degree in computer sciences from Purdue Univ., USA. He is professor and Chair in Digital Health, School of Computing and Comm., and Div. of Health Research, Lancaster Univ., UK. Before joining Lancaster he was professor in the dept. of Computer & Info. Science and Eng., Univ. of Florida, USA, where he directed the Mobile and Pervasive Computing Lab. His research spans pervasive systems, IoT and digital health.

Ahmed E. Khaled received the Ph.D. degree in computer science from Univ. of Florida, in 2018. He is currently assistant professor, computer science dept., NE Illinois Univ., USA. He received the B.Sc. (2011) and M.Sc. degrees (2013) in computer eng. from Cairo Univ., Egypt. His research interests span IoT, smart spaces, and ubiquitous computing.

Wesley Hutchinson received a BSc in Molecular Medicine (2004) and MB BS (2007) from Univ. College London; MRCGP (2012); PGDip [Diabetes] (2016); PGCert [Computer Science] (2018). Currently a GP Academic Training Fellow at Lancaster Univ. pursuing a PhD in Digital Health, whilst continuing his clinical work as a GP. His research focus is the application of digital health assistive technologies for clinicians. He is a member of the Royal College of General Practitioners and the IEEE Computer Society.