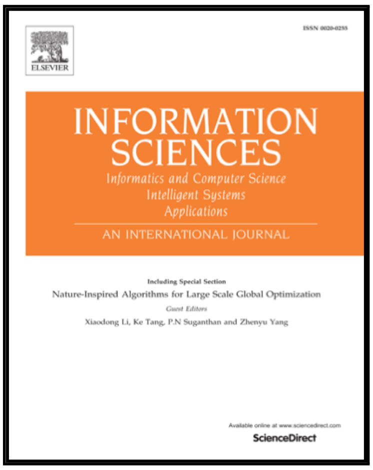

Modelling Trust Dynamics in the Internet of Things

Carmen Fernandez-Gago, Francisco Moyano, Javier Lopez

PII: S0020-0255(17)30536-4
DOI: 10.1016/j.ins.2017.02.039
Reference: INS 12762

To appear in: *Information Sciences*

Received date: 16 November 2015
Revised date: 1 December 2016
Accepted date: 15 February 2017

Modelling Trust Dynamics in the Internet of Things

Carmen Fernandez-Gago[a], Francisco Moyano[a], Javier Lopez[a]

[a]*Network, Information and Computer Security Lab*
University of Malaga, 29071 Malaga, Spain

Abstract

The Internet of Things (IoT) is a paradigm based on the interconnection of everyday objects. It is expected that the 'things' involved in the IoT paradigm will have to interact with each other, often in uncertain conditions. It is therefore of paramount importance for the success of IoT that there are mechanisms in place that help overcome the lack of certainty. Trust can help achieve this goal. In this paper, we introduce a framework that assists developers in including trust in IoT scenarios. This framework takes into account trust, privacy and identity requirements as well as other functional requirements derived from IoT scenarios to provide the different services that allow the inclusion of trust in the IoT.

Keywords: Trust, Internet of Things, Dynamic framework

1. Introduction

The Internet of Things (IoT) is a paradigm based on the interconnection of everyday objects. According to the Gartner report for 2013 [3], 26 billion objects are expected to be connected in the IoT by 2020. From an economic perspective, the same report also highlights that IoT is expected to generate $1.9 trillion from the production of IoT products and service suppliers, which will translate into economic growth and employment. At the same time the amount of data managed in the IoT makes it necessary to look at the data-centric perspective [24] and consider the privacy implications that this might raise. The advantages brought by IoT could be seriously threatened if the reception from society is negative. This could be a possibility if citizens, companies and administrations feel that they cannot trust the IoT. Users are becoming more aware of the importance of protecting their private information [2, 14], and companies are increasingly realising that an incorrect security strategy can lead to important economic and reputation losses, and eventually, to bankruptcy[1].

Email addresses: mcgago@lcc.uma.es (Carmen Fernandez-Gago), moyano@lcc.uma.es (Francisco Moyano), jlm@lcc.uma.es (Javier Lopez)

[1] http://www.pcworld.com/article/2046300/hackers-put-a-bulls-eye-on-small-business.html

Gartner's report for 2014 stresses that companies have realised this and, as a result, in 2014, increased their investments in security by around 8% [1]. But even if IoT systems are actually secure, society may be reluctant to use them if trust concerns are not appropriately addressed.

It is a fact that things will have to interact to generate business value. Interactions will often have to happen in uncertain conditions. Having mechanisms in place that help the 'things' involved in IoT scenarios overcome the lack of certainty becomes of paramount importance. Traditional security mechanisms are not enough; however trust management systems can help in these cases. They provide a greater flexibility than traditional security mechanisms, easing the decision-making process. In the end, engineering these trust concerns in IoT services and systems must be a primary goal to ensure the successful adoption of the IoT paradigm. There are several related challenges that must be overcome. First, from a technical point of view, the IoT itself brings about new challenges concerning security and trust, given that new interaction models, such as Machine to Machine (M2M), are gaining traction. Second, the nature of IoT scenarios make them higly dynamic and heterogeneous as things are constantly leaving and entering the IoT environments. Thus, the systems designed for IoT environments should reflect these challenges and should take into account the following:

- *Interoperability.* Devices with different capabilities from different manufacturers and, probably adhering to different standards, must be able to communicate. Moreover, the different trust management systems that may co-exist in IoT environments have to be interoperable and able to exchange information from other trust systems.

- *Dynamicity.* The dynamicity of IoT systems, where new devices and services may enter and leave the system at non-predictable intervals, implies that trust management systems must also evolve with the systems.

- *Fragmented research.* The previous issues are being tackled by their research communities in isolation. This is also the case in other important research areas that must support trust in IoT systems, such as identity management and privacy. A holistic approach is needed.

To summarise, the complexity of IoT technologies and the fragmentation in IoT research are two stumbling blocks that prevent developers from gaining the adequate know-how to design and implement full-fledged IoT systems that can be trusted. Consequently, we can expect that the systems built will suffer and that end-users will not be satisfied. We advocate that better tools can build better products. We introduce a framework that comprises a set of tools and services that designers and developers can use to integrate trust concerns into IoT systems. The framework targets designers and developers, but its benefits will be reflected in the end-users of IoT systems, who will feel more confident about its adoption as they will eventually have a better quality experience.

The paper is structured as follows. Section 2 reviews existing work on Trust Management for the IoT. Section 3 delves into the problem of trust and the IoT, and Section 4 describes our proposal of an architecture for including trust in IoT systems. Section 5 demonstrates how the framework can be applied in an IoT scenario. Finally, Section 6 concludes the paper and outlines future work.

2. Related Work

The concept of trust in computer cience is taken from the concept in sociological, psychological and economical environments. The definition of trust is not unique. It may vary depending on the context where, and what purpose it is going to be used. Despite being seen as of paramount importance when considering systems security, a standard definition of trust has yet to be provided. However, it is widely accepted that trust might assist decision-making processes such as those involved in access control schemes.

Trust management systems first emerged in the literature as a way of solving access control problems and unifying authentication and authorisation in distributed systems [8]. The origins of computational trust date back to the nineties, when the work in [15] analysed social and psychological factors that have an influence on trust and replicated this concept in a computational setting. Since then, many different trust management systems have been developed for different applications. A trust model comprises the set of rules and languages needed to forge trust among entities in an automatic or semi-automatic way.

The heterogeneity in the number of trust management systems often leads to confusion as one might easily lose the most relevant concepts that underpin these trust models. By trust concept or trust-related concept, we refer to any notion that has a high relevance according to how frequently the notion arises in existing trust models. By analysing these trust concepts, the authors in [17] designed a conceptual model for trust that serves as the basis for a development framework that supports the accommodation of heterogeneous trust and reputation models [19]. In this approach, the authors distinguished two types of trust models:

- *Decision models.* Trust management has its origins in these models [8]. They aim to make access control decisions more flexible, simplifying the two-step authentication and authorisation process into a one-step trust decision. Policy models and negotiation models fall into this category. They build on the notions of policies and credentials, restricting the access to resources by means of policies that specify which credentials are required to access them.

- *Evaluation models.* These models are often referred to as computational trust, which has its origin in the work in [15]. Their intent is to evaluate the reliability (or other similar attribute) of an entity by measuring certain factors that have an influence on trust. Two sub-types of models in this category are propagation models, which disseminate trust information along trust chains, and reputation models, in which entities use the opinions of others about a given entity in order to evaluate their trust in the latter.

There is very little effort being made to design trust management systems for the IoT. A specific IoT environment where the 'things' are only wireless sensors is considerd in [9]. The trust management solution in this case only solves the problem of packet forwarding. This approach therefore does not deal with the heterogeneity that the IoT paradigm targets. The authors in [6] designed a scalable trust management protocol for IoT that takes into account social relationships and uses properties such as honesty, cooperativeness and community interest in order to evaluate trust. The protocol is distributed and the nodes update trust only for the nodes they are interested in or interact with. The updates are done through direct observations or indirect recommendations. Based on this model, the same authors proposed a dynamic trust management protocol for the IoT to deal with misbehaving nodes or behaviour that may change dynamically [5]. A different point of view on how the things interact in the IoT paradigm is presented in [4]. In this paper the authors considered that the objects in an IoT scenario conform a social network where they establish social trust relationships. They introduced an architecture for the Social Internet of Things (SIoT). Trust is not explicitly considered but they propose a method for determining trustworthy nodes in this socialised environment. The work in [25] proposed a centralised trust management system for the IoT that aims at managing cooperation among nodes with different resource capabilities. The model assigned trust values to cooperating nodes according to different contexts. None of these approaches consider the inclusion of trust in IoT environments in a dynamic way by considering it in the early stages of designing IoT services as we propose in this paper. In order to deal with dynamicity in the IoT, we introduce the concept of trust@run.time (see Section 3). There is a growing interest in considering notions of trust in self-adaptive systems in order to leverage reconfiguration decisions, especially in the areas of multiagent [13, 26], component-based [12, 27] and service-oriented systems [11, 23]. These approaches advocate the use of trust and reputation to evolve highly dynamic and security-sensitive systems, which justifies its exploration in broader use cases such as those present in the IoT.

3. Challenges for Integrating Trust in the Internet of Things

'Things' in IoT environments are expected to interact with each other. In most cases the interactions will have to happen even if there is not enough information about the things to establish them. The information available about a thing in an application might not only come from its behaviour from others' interactions with it but also from the information that may be provided due to all the 'things' surrounding it. Our assumption is that things are not just physical entities but rahter the whole set of 'things' that interact with them; this is what we call the *context*. Figure 1 illustrates a traditional IoT scenario where the context of the person in the figure (a thing as well according to our assumption) is depicted by the objects to which the arrows are pointing.

There are two main challenges that we need to address if we wish to provide a holistic solution to trust management in the IoT: interoperability and dynamici-

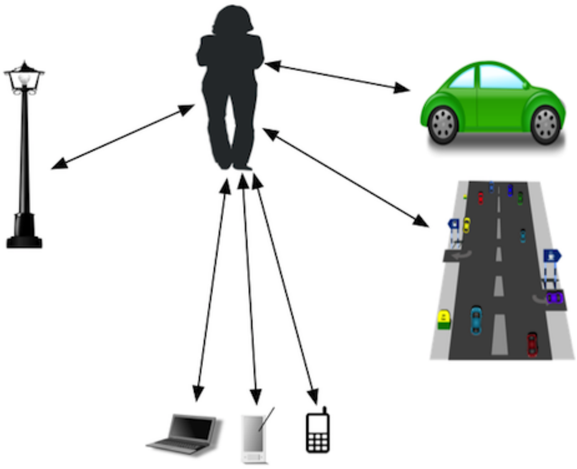

Figure 1: A Context for a User

ty/evolution. Interoperability is a problem that is derived directly from the fact of dealing with the heterogeneity of the IoT. Different things will have their own trust management systems that will have to exchange information with others, and thus different trust management systems that may coexist. The proposed framework will enable trust models using different languages or different ways to determine trust to derive common trust information for all.

In terms of system evolution, there exists a bidirectional relationship between IoT systems and their trust management systems. On the one hand, given that IoT systems and their contexts are dynamic, the underlying trust management systems must change to meet the most recent trust concerns. For example, new sources of trust-related information may appear. On the other hand, trust and reputation values can lead to changes in the IoT systems. If the trust relationship between two 'things' falls below a certain threshold, or the reputation of a 'thing' is too low, changes in the system may be required to maintain a tolerable level of trust. The dynamicity of such scenarios and the building of trust management systems that change at runtime have so far been left out of the literature. We propose considering 'trust@run.time', which has been developed from the concept of models@run.time [7]. This term refers to maintaining an abstract model of the executing system in such a way that both are always synchronised. This pushes the idea of reflection one step further as we can reason about the running system in terms of the model. This idea has become widely accepted among the self-adaptive community, as it proposes that changes in the model are automatically reflected in the running system, encouraging a fast and fluid evolution. We advocate this as a natural step towards supporting high dynamic IoT systems, because different trust and reputation models may be required depending on the contexts of the systems over their lifetime. The idea of trust@run.time was first proposed by [16], but there is a growing interest in considering notions of trust in self-adaptive systems in order to leverage reconfiguration decisions.

4. Including Trust in the IoT

The framework we propose aims to assist developers when adding trust or reputation in IoT systems. Instead of having to implement each trust model from scratch, our framework facilitates the work of the developers by providing them with techniques and guidance for re-using common features of other trust models and following certain steps to carry out the implementation.

Trust and reputation requirements are not the only ones affecting IoT scenarios. Privacy and identity management, as well as other non-functional requirements, will be of paramount importance in building the framework. If we are interested in developing a framework where different trust management systems for the IoT are present, we need to consider aspects of identity management. It is particulary crucial to properly define the identity of the 'things'. Their identity could be determined by their context (the set of things that are connected with the user for a specific purpose at a given moment in time). It may not be enough, or even advisable for things to identify themselves by providing some kind of identification, for example, a login. The identity of a thing may vary depending on the context where it is set. Privacy, is the other important pillar when modelling trust. There is a direct relationship between them. In some cases, it could be that the more information is disclosed, the greater the accuracy of the trust-based decision. In turn, information disclosure raises privacy concerns that need to be taken into account. However, it could be that trust helps to preserve privacy as it can be used to prevent establishing communication with an untrustworthy thing.

Let us look at the thing, a person in this case, in Figure 1. Her useful identity for this scenario is the one that, at the time she is passing through a road, can obtain from or offer information to the other things around her (lampost, card, etc.). It is not relevant for this scenario whether she is a doctor or a lawyer, for example. These two features might be relevant in other scenarios, for instance, when she is dealing with the tax or national insurance offices in her country.

Keeping all these considerations in mind, we believe that both privacy and identity are properties that should always be present when designing trust and reputation management systems.

4.1. Architecture

In order to build the framework for the development of trust management systems for IoT, the architecture that we propose is divided into four layers, where each of them builds upon the outcomes of its lower layer. The layers are the following:

- *Scenarios Layer.* This layer deals with the identification of IoT scenarios. From these scenarios, the different contexts that may arise in each of them are identified. Since dynamicity and evolution are captured by the changes in the environment at each moment in time, our intention is to capture these changes at the context level. In this layer we set up the basis for a key concept that we introduce: the concept of context of a thing.

- *Requirements Layer.* The contexts identified in the preceding layer will be the basis for deriving requirements related to identity management, privacy and trust in this one. These requirements will be used in different elements of the services layer. There could be several ways to represent requirements. In [20] an extension of UML with trust requirements is used to represent requirements that may arise when designing a trust model. We will use an approach based on SI* where an extension of trust to include the representation of trust requirements [22] is presented. We will use this extension to include privacy and identity requirements. Exactly how this extension is going to be done is beyond of the scope of this paper for reasons of length.

- *Services Layer.* The services included here range from the definition or storage of contexts, to the implementation of the trust models, how to consider interoperability or how dynamicity and evolution is dealt with. Given that trust models must evolve alongside the system, trust requirements influence dynamicity. The dynamicity and evolution service will allow developers to access and modify the trust models.

- Trust Framework Layer. This layer is the realisation and ultimate goal of the framework. It includes several services that are packaged into a workflow-oriented development framework that is ultimately delivered to designers and developers through APIs (Application Programming Interfaces). The framework consists of an API for developers with some base components that can be extended, some methods that can be overridden, and configuration files.

Figure 2 shows the proposed architecture, built in a bottom-up approach manner.

4.2. Trust Framework and Services

In this section, we concentrate on the core layer of the framework, which is the services layer that will result in the proposed framework. The scenarios and requirements layers are used as the inputs for the service layer. They are determined by the different use cases and the requirements identified for them.

4.2.1. Context Definition

The dynamicity of a trust model could be captured if we are able to determine the factors that influence trust in a given moment in time for a specific purpose. We advocate that trust models that are going to be defined for a given 'thing' do not depend only on its behaviour but also on what we call the *Context* or things around it. Contexts refer to a specific moment in time. Thus, we

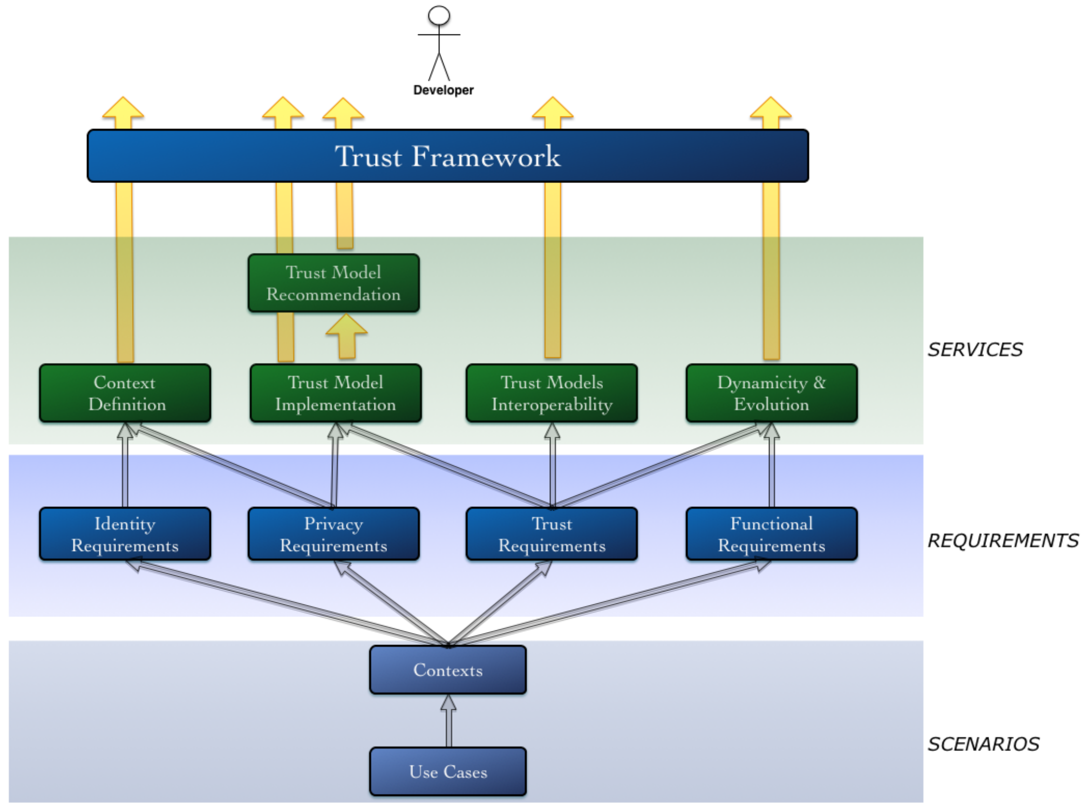

Figure 2: Architecture of the Trust Framework for IoT Environments

define the context of a thing as the set of things that are connected to it (including the thing we are defining the context for) in a given moment in time and that provide information for a specific purpose. These purposes influence their behaviour and their relationship with other things. They can be, for instance, sensing temperature, measuring traffic, and so forth.

Due to the heterogeneity of the IoT scenarios, there might be a huge number of contexts even for a single use case. The identification of the different contexts will give us information about the requirements posed in the IoT, in particular, those related to identity, privacy or trust.

The Context Definition service will rely on the set of things ($\mathcal{T}$) of a given scenario. For a specific purpose, p, and a given thing (T_i), the service will provide a subset of $\mathcal{T}$ at a specific moment in time, t_i. In addition, the mere fact of considering the evolution of the scenario over time may influence the existence of contexts. The purpose determines the subset of things that are connected to T_i and have to be considered. Thus, for instance, if the purpose is to have information about the state of the traffic, it is irrelevant to consider the things that are controlling domestic electricity consumption. The service will also be provided with a repository of the contexts that can be used to define new contexts by re-using existing ones. This will make defining trust models easier as the actors (trustor and trustee) do not always have to be defined as new and they can be re-used from other contexts.

4.2.2. Interoperability among Trust Models

One of the main features of IoT environments is the variety of things that are interconected. This means that different trust models might have to interact even if they are of different types. Thus, it would be desirable for them to be able to interpret and understand each others languages and their ways of deriving trust. Achieving interoperability can be done by the establishment of mappings between the models, which are usually mappings between different mathematical functions. In the case that the models interacting are both evaluation models (those that compute trust by using a trust engine that derives a concrete value, such as in the case of reputation models), they should use the same scales as the others they have to interact with, the same aspects to measure or resulting trust values. Thus, the mapping to be defined in these cases will be a mapping that allows both models to use the same scales and interpret the results in the same way. Mappings between evaluation and decision (policy-based) trust models are a more complicated issue that will have to be done as well. Since these two types of models work in different domains, the problem can be tackled by establishing common semantics and from this point on, define mappings between the different trust values of each model, whatever the format. As an example of very naive trust models we describe the following situation. Let us imagine an evaluation model where trust outputs are 0, 1, 1.5 and 2. Let us also imagine a very simple decision model where the outputs provided are *trust is established* and *trust is not established*. A very straighforward way of mapping these models would be to say that 0 and 1 could mean *trust is established* and 1.5 and 2 could mean *trust is not established*. The interoperability service should therefore allow the precise definition of these mappings. The definition of the mappings is not an easy issue and cannot be generalised as it will depend on the different scenarios, which have different requirements and trust models.

4.2.3. Dynamicity and Evolution Service

IoT scenarios are inherently dynamic, leading to changes in the environment and the contexts that live within them. Trust management systems must adapt themselves at runtime as a response to these changes. In order to tackle dynamicity, it is necessary to represent the current state of the system and its trust relationships at runtime. This service allows the system to be changed in accordance with changes in the trust relationships or reputation values. One way to tackle the problem of dynamicity is the concept of trust@run.time (Section 3), where there is a representation (a model) of the trust management system, which is synchronized with the actual running systems. This allows us to reason about the system and perform high-level changes that are automatically translated into changes of the running system.

The functionalities offered by the service can be divided into two large areas: trust@run.time and reconfiguration policies specification. The former is a paradigm introduced by Moyano et al. [16], which is a natural evolution over the models@run.time [7]. In essence, we count on a reflection layer that represents the state of the system and the trust models at runtime with models. Any change to these models entails an adaptation of the system to comply with the new model.

4.2.4. Implementation Service

The dynamic framework that we are proposing should also include guidelines for developers to implement the trust models. We can use the implementation framework presented in [16, 18]. This framework consists of several classes that developers can customise via inheritance and by overriding or implementing some of their methods. These classes use the Kevoree framework [2] [10] classes as the fundamental building blocks. Developers can build trust and reputation models right into the models@run.time platform provided by Kevoree, which in turn enables trust and reputation information to be used for reconfiguration decisions at runtime.

4.2.5. Trust Model Recommedation Service

This service is not strictly needed for the framework but it appears as a result of the other services and it can be useful for finding the most appropriate trust model for each case. In a heterogeneous environment like the IoT, it may sometimes be highly useful to choose the most approporiate trust model to use from among the different ones available for the same thing.

The next section provides further insight into the framework by demonstrating its application in a real scenario.

5. Application Scenario: Field Service Teams

The scenario that we have chosen to show how the framework presented in Section 4 could be applied, considers a field service team (FST)[3]. There is a system, which we call the Dispatching System (DS), which allocates tasks to operators. This allocation process requires a decision, and this decision can be supported by trust. The goal of the DS is to optimise allocations by assigning tasks to those operators who can be more trusted the most to fulfill these tasks. A task may involve several factors, including an estimate complexity, a safety/risk level, an estimated duration and the preferred/required professional profile. Likewise, operators have several factors that the DS may exploit, like their professional profile, the total working hours up to that moment, the tasks completed during the day, the proximity to the location of the task, the years

[2] http://kevoree.org

[3] `http://community.dynamics.com/b/msftdynamicsblog/archive/2015/04/10/the-intelligent-service-technician-empowering-field-service-with-smartglasses`

of experience and a reputation score. Most of this information can be gathered from the context of the operators, which consists of devices carried by them, such as smart glasses, PDAs/tablets, and smart watches. This context may change dynamically due to different events: an operator forgets the PDA/tablet, an operator switches from the PDA to a smart watch, the wireless connection between the system and the tablet is cut off due to problems with the device, etc. Also, the operator may need a car to get to the place where the task is to be performed; therefore the car would be added to the context of the operator, and could provide further information that the DS can exploit to make a trust decision, such as the remaining petrol in the tank (e.g. the DS cannot trust an operator to complete a task if the operator cannot reach it before the finish time). The system must be able to dynamically adapt to these changes in the contexts of the operator in order to maximise the information gathered and its utility before making a trust decision.

5.1. Use Case

Imagine a gas company. A sensor has detected that a pipe is leaking gas and informs the Problem Detection System (PDS) of the location of the problem and its criticality, which is set to high. The PDS, querying an internal Experience Database, estimates the duration of the task and the most suitable professional profile, and forwards all this information to the DS.

The DS initiates probes with all the things connected to the corporate network. Anne's smartphone receives a probe and sends an *ack* back to the DS. The DS requests further information, such as Anne's location, the name of the smartphone's owner (i.e. Anne) and the total work hours in the day. The smartphone sends GPS and the contact information of Anne back to the DS, but given that it ignores the total work hours, it looks up other devices in the same context until it finds Anne's PDA. The latter controls the total hours of work and sends it to the smartphone, which in turn forwards it back to the DS. Given that the location of the task is a 20 minute walk away, the DS sends the location to Anne's car, which is in her context too, and which calculates whether the car has enough petrol to get from its current location to the task's location. In this case, there is enough, and therefore the car sends a positive answer back to the DS. With all this information, the DS computes a trust value that turns out to be the highest of all operators in the surrounding area. Therefore, the DS sends Anne's PDA the new task assignment, but there is a connection problem and so the DS sends it again but this time to an app installed on the smartphone. A new task assignment pops up on the app providing Anne with information about the location of the task. Upon arriving at the location, Anne requests further information about the structure and material of the pipe, and the Task Visualisation System sends such information as a Heads-Up Display interface to her smart glasses.

The same process is done with another user, Bob. Thus, the gas company will have information from both users in order to be able to make a decision as to who is the best one to go to and solve the problem of the leaking pipe.

5.2. Application of the Framework

The framework that we have introduced in Section 4 can be used to implement the scenario (i.e. the framework is to be used by software engineers, and not by the gas company). As we have seen in Section 4, the framework is composed of different layers. In the following paragraphs, we will detail how this framework can be applied in the use case presented in Section 5.1, layer by layer. We will exemplify it in the case of operator Anne but, as we have mentioned, the process will be similar for each operator in the field.

5.2.1. Scenarios Layer

The bottom layer of the framework (as depicted in Figure 2) defines the use cases and identifies the contexts involved in each of them. For the sake of simplicity for the description, we are going to identify only two contexts for the use case described above, and only in the case of user Anne.

At the initial moment in time, t_0, let us assume that the context of user Anne comprises a PDA, a smart watch and smart glasses. The purpose, p, we consider is the assignment of the task 'attending to a leaking pipe' by the gas company. Thus, the context of user Anne (T_0) is defined as $C_0 \equiv C_{T_0}^{t_0}(p)$ and contains the elements PDA, smart watch and smart glasses. As inferred from the definition of context in Section 4, the user is also part of her own context as one more thing in the whole set.

These things in the context of Anne will help gather information about her proximity to the task, the remaining hours that Anne can work on assigned tasks, the number of tasks completed and the reputation of Anne, which is updated by a reputation model (run by the gas company) once Anne has finished previously assigned tasks.

Let us assume that in a different moment in time, Anne leaves her PDA and uses a car. Then, her context changes, being in this case, $C_1 \equiv C_{T_0}^{t_1}(p)$ composed of a smart watch, smart glasses and a car (assumming car has wireless connection). The information that the context of Anne will be gathering and providing will be pretty much the same, although adding in this instance, information about the petrol in the tank needed to reach the place of the leak.

It is expected that this service will maintain records of all the contexts in order to re-use them when a new one is received. The idea is to re-use information about things and not have to define them all from the scratch.

5.2.2. Requirements Layer

This layer will be responsible for gathering the requirements derived from each of the contexts for each use case. As shown in Figure 2, the requirements are clustered in different categories. We are however interested in providing more insight into trust requirements, given that the main purpose of the framework is to include trust, although functional requirements should be taken into account when designing the whole system. Privacy and identity requirements are also of paramount importance as we consider they influence trust. Functional requirements are beyond the scope of this paper.

Identity requirements. One of the main features of our framework is that it is based on dynamicity. This dynamicity is derived from the scenario and the framework supports the implementation of such dynamic scenarios. Dynamicity is reflected in the different identities that a thing could have, depending on the context where it is. There should be an identity management system that assigns an identity that it is context-dependent to each of the things. Thus, for instance, the identity of the smart watch that Anne is wearing in $C^{t_0}_{T_0}(p)$ should be different from the one she is wearing in context $C^{t_1}_{T_0}(p)$, as they may be gathering different information because they are capturing different moments in time.

Privacy Requirements. It is very important that the information that the company and Anne exchange, remains between them and is not displayed to another employee. This is the main privacy requirement that we foresee for the scenario that we are considering.

Trust Requirements. The ultimate goal of the gas company is to have all the information available to make a decision as to the most suitable employee to solve an incident, that is, who is most trusted by the company to accomplish the task. To determine this, each of the things of each of the contexts provides information to the gas company. This information is based on different factors and it is processed by the gas company, using a trust engine for each of the things. The trust engines are essential parts of the trust management systems of the things. We consider that there is a trust relationship between each thing and the company that it is handled by a trust management model for each of them. These trust models help determine the reliability of the information passed from the thing to the company. Therefore, another trust requirement is how the relationship between the thing and the company is established. This relationship should rely on different factors that the company establishes, for instance, if the thing is measuring distance to the point of the task, the company should consider how accurate this distance on other occasions was. Other examples include how often the things had to be fixed in the past, the time of the last supervision, whether the thing's firmware or operating system have any security certification, etc. All the trust relationships in the system are depicted in Figure 3. This figure shows the scenario described in Section 5.1 with two contexts for Anne, at two different moments in time. It also describes one context for Bob.

The company will also have a reputation system that evaluates the performance of Anne with respect to the tasks she has been assigned to. Thus, there must be a way to provide an *a posteriori* feedback about how Anne performed. This feedback may be from her supervisors or from the sensors themselves once they measure whether the pipe has been completely fixed or not. This reputation value will be another input for the trust engines for the decision-making process. Thus, the final trust decision will be made by considering and combining the values obtained from the different aforementiones trust engines and the reputation system.

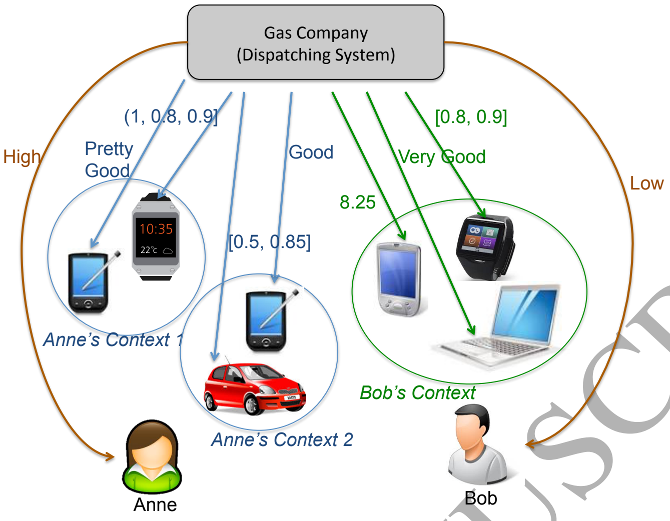

Figure 3: Trust Relationships in the FST scenario

The Dispatching System establishes a trust relationship with each thing, which measures the reliability of the information passed on by it. These trust relationships can use different trust models, and thus different trust engines, which provide the output of trust values in different formats, such as intervals or qualitative labels; hence the importance of providing interoperability support. Likewise, the Dispatching System establishes trust relationships with the operators with respect to a pending task.

5.2.3. Services Layer

This is the layer where the services that are offered to developers reside. Each service has a very specific task and tackles the concerns discussed in Section 3. We now explain further how these services will work.

Context definition. This service will implement the different contexts that are present in each scenario. For the case we are considering, the two contexts we have identified have several elements in common. Thus, the context definition service will serve as a kind of a database for all the contexts and will re-use information on their components, as in some cases (as occurs in our case with C_0 and C_1), there are common elements.

As part of the context definition, potential devices and their capabilities must be modelled, contexts must be given unique identifiers and strategies must be provided to detect and react to transitions between contexts. As part of the service, several primitives should be included, such as a function that can determine all the things that are part of a given context, at a given moment in time. The service should also track the different contexts that exist over the lifetime of the system (up to a limited amount of backup memory).

In our case, developers should model a plethora of devices, including smartphones, smartglasses and the on-board circuits of the car. Developers should also model the human things that are part of the contexts, such as Anne and Bob (i.e. operators in the general sense). The service should allow developers to express ownership relationships (e.g. Anne has a car), information sharing relationships (e.g. Anne's car is sharing information about Anne) and purpose information (e.g. Anne's car is sharing information about Anne in order to fulfill goal *fix the pipe*). With all this information, contexts and changes in contexts can be deduced automatically without the need for manual updates. Whenever a new context is detected, a new unique identifier is generated for this new context, and a parent-child relationship is created to represent the transition between contexts.

Trust Models Implementation. The use case that we are analysing has different trust or reputation models that need to be implemented. This service allows the identification and implementation of all the different building blocks that constitute a trust or reputation model, according to the methodology described in [18, 21].

The service defines the trust entities and their trust relationships, which in our case means Anne, Bob, the DS and every thing that enters the scene. Likewise, the service defines which entities can rate which other entities. In this case, there may be sensors and supervisors that can act as sources of reputation about Alice and Bob (i.e. operators).

The service also supports the creation of trust and reputation engines. Thus, developers can specify which factors (i.e. inputs) the engine accepts, how these factors can be updated, and how they are combined to yield a trust or reputation score.

As for our case study, we have seen that each thing can hold a trust relationship with the dispatching system. Developers could implement different models based on the capabilities of the things or the information available to them. As an example, let us consider the trust relationship between the DS and the corporate PDA of Anne. The first step for the developers should be to model the dispatching system and the PDA as trust entities. Then, developers should think about how the trust model computes the trust value of its trust relationships. In our example, consider that the PDA is checked every week by the IT team of the company. The IT team provides a report on potential problems or vulnerabilities detected in the PDA. This report becomes an objective factor that developers can use as input for the trust engine of the model. Therefore, developers could implement a trust model where trust is computed by averaging all the fields of the report. As part of the model, developers could also add a trust threshold computation. Thus, the dispatching system could make trust decisions depending on whether the trust value of the trust relationship with the PDA is above or below the computed threshold.

Trust Models Recommendation. This service is built on top of the Trust Models Implementation service and provides developers with templates for already-existing, well-known trust models, including the trust engines and the factors used by those engines. Developers can modify this template or complete it with further information, like the specific instances that will represent the trust and reputation entities (e.g. Anne, Bob and supervisors).

In our example, we consider that developers wish to apply Marsh's model [15] for the trust values and to compute threshold values between the dispatching service and Anne's smartphone. Therefore, developers can choose the model from a pull-down menu, and this choice generates all the configuration files, data structures and classes (in Object-Oriented terms) required to implement the model. In particular, the DS and Anne's smartphone become trust entities, and the trust engine accepts as inputs, the utility of the collaboration, the importance of the collaboration and the general trust (trust based on the history of interactions). The output is a real value resulting from multiplying these factors. Developers should instantiate these factors, meaning that they should relate these factors to scenario-specific sources of information. Likewise, the template generates a function for the computation of the threshold, which takes as inputs the perceived risk, the perceived competence, the importance of the collaboration and the general trust.

Trust Models Interoperability. In accordance with the trust requirements above, the gas company uses the different trust engines belonging to the different trust models from the different things to process the information they provide. It is likely that the outputs from these engines take different forms. For example, Marsh's model yields real values in the interval $[0, 1]$, but other models may yield trust values in different formats, including discrete numbers or qualitative labels like bad or good.

The way the DS processes the trust values obtained from the contexts can be done in different ways. Depending on the scenario, the DS could either obtain a global trust value that it derives somehow from all the things in all the contexts, or it might compute a trust value for each of the contexts. If the choice is to have a global trust value for each context, the DS may need to generate a unique value from different sources. Thus, for example, in the case of context 2 of Anne in Figure 3, the trust outputs from the car are in the form of numeric values whereas the trust outputs from the PDA are in terms of qualitative attributes such as *Good*. In this case, if we wish to combine these two different outputs, the trust models' interoperability service should provide a way to do so, either by mapping from qualitative values to numeric ones or vice-versa.

In summary, this service allows developers to define mapping policies to translate the semantics of one model into the semantics of another. One way to accomplish this would be to attach meta-data to the models, either as part of annotations in code or as XML/JSON tags in configuration files. The meta-data of a trust model includes the possible maximum and minimum values of the model, whether a minimum value means distrust or simply a lack of information, or whether complete trust is ever achievable. Whenever there is a new collaboration between the dispatching system and a thing, this metadata is sent together with the rest of information, and the system can therefore store this information and use it whenever it needs to translate from one model to another.

Dynamicity & Evolution. The specification of reconfiguration policies refers to the specification of the conditions in which the system must be changed. For example, developers can specify that if the trust value of the trust relationship between the dispatching system and Anne's smartphone falls below the threshold specified by the trust model (e.g. Marsh's model, as discussed earlier), the communication interface should be cut off and the dispatching system should look for other, more trusted things in Anne's context from which to retrieve the required information.

6. Conclusion

The rise of the IoT paradigm is bringing with it new challenges concerning security and trust. In this paper, we have discussed the challenges inherent in trust that have to be overcome for IoT scenarios and have introduced a framework to be used by developers to include trust concerns in IoT systems. The architecture of the framework comprises different layers, where the upper layers depend on the bottom ones. The framework ensures that trust is included in all the phases of the development of IoT systems following a proactive approach, as opposed to an afterthought service, which has been the standard in tackling trust until now. The key points of the framework are the considerations of the triad comprising trust, identity and privacy requirements and the possibility of taking into account dynamicity and evolution.

We have described a scenario and a use case that show an exemplification of the IoT and how the framework is applied to it. It remains for the future to work on an implementation of the framework, which will principally concern the implementation of all the intermediate layers and services that are part of it. We have provided some hints as to how the implementation can be approached for each service. An initial step towards this is the inclusion of privacy and identity requirements into existing requirements specification languages defined for trust, such as those based on SI*.

Acknowledgements

This research has been partially supported by the Spanish Ministry of Economy and FEDER trhough the projects PRECISE (TIN2014-54427-JIN) and PERSIST (TIN2013-41739-R), and the European Training Network NeCS (H2020-MSCA-ITN-2015- 675320).

References

[1] Gartner newsroom, http://www.gartner.com/newsroom/id/2828722.

[2] Are Facebook Users more Privacy Aware Now? (2012).

[3] The Gartner report, `http://www.gartner.com/imagesrv/pdf/Gartner_2013_annual_report.pdf` (2013).

[4] L. Atzori, A. Iera, G. Morabito, M. Nitti, The social internet of things (siot) – when social networks meet the internet of things: Concept, architecture and network characterization, Computer Networks 56 (2012) 3594–3608.

[5] F. Bao, I.-R. Chen, Dynamic trust management for internet of things applications, in: International Workshop on Self-aware Internet of Things, ACM, 2012, pp. 1–6.

[6] F. Bao, I.-R. Chen, Trust management for the internet of things and its application to service composition, in: IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE, 2012, pp. 1–6.

[7] G. Blair, N. Bencomo, R. B. France, Models@ run.time, Computer 42 (10) (2009) 22–27.

[8] M. Blaze, J. Feigenbaum, J. Lacy, Decentralized Trust Management, in: Proceedings of the 1996 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, 1996, pp. 164–173.

[9] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, X. Wang, Trm-iot: A trust management model based on fuzzy reputation for internet of things, Computer Science and Information Systems 8 (4) (2011) 1207–1228.

[10] F. Fouquet, O. Barais, N. Plouzeau, J.-M. Jézéquel, B. Morin, F. Fleurey, A Dynamic Component Model for Cyber Physical Systems, in: 15th International ACM SIGSOFT Symposium on Component Based Software Engineering, Bertinoro, Italy, 2012.

[11] H. Hanen, J. Bourcier, Dependability-Driven Runtime Management of Service Oriented Architectures, in: PESOS - 4th International Workshop on Principles of Engineering Service-Oriented Systems - 2012, Zurich, Switzerland, 2012.

[12] P. Herrmann, H. Krumm, Trust-adapted enforcement of security policies in distributed component-structured applications, in: Sixth IEEE Symposium on Computers and Communications, 2001, pp. 2–8.

[13] L. Klejnowski, Y. Bernard, J. Hähner, C. Müller-Schloer, An Architecture for Trust-Adaptive Agents, in: 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), IEEE, 2010, pp. 178–183.

[14] M. Li, X. Sun, H. Wang, Y. Zhang, J. Zhang, Privacy-aware access control with trust management in web service, World Wide Web 14 (4) (2011) 407–430.

[15] S. Marsh, Formalising trust as a computational concept, Ph.D. thesis, University of Stirling (April 1994).

[16] F. Moyano, Trust engineering framework for software services, Ph.D. thesis, University of Malaga (2015).

[17] F. Moyano, C. Fernandez-Gago, J. Lopez, A conceptual framework for trust models, in: S. Fischer-Hübner, S. Katsikas, G. Quirchmayr (eds.), 9th International Conference on Trust, Privacy & Security in Digital Business (TrustBus 2012), vol. 7449 of Lectures Notes in Computer Science, Springer Verlag, Springer Verlag, Vienna, 2012, pp. 93–104.

[18] F. Moyano, C. Fernandez-Gago, J. Lopez, Building trust and reputation in: A development framework for trust models implementation, in: A. Jøsang, P. Samarati, M. Petrocchi (eds.), 8th International Workshop on Security and Trust Management (STM 2012), vol. 7783 of LNCS, Springer, Springer, Pisa, 2013, pp. 113–128.

[19] F. Moyano, C. Fernandez-Gago, J. Lopez, A framework for enabling trust requirements in social cloud applications, Requirements Engineering 18 (2013) 321–341.

[20] F. Moyano, C. Fernandez-Gago, J. Lopez, Towards engineering trust-aware future internet systems, in: X. Franch, P. Soffer (eds.), 3rd International Workshop on Information Systems Security Engineering (WISSE 2013), LNBIP, Springer-Verlag, Valencia, Spain, 2013, pp. 490–501.

[21] F. Moyano, C. Fernandez-Gago, J. Lopez, A model-driven approach for engineering trust and reputation into software services, Journal of Network and Computer Applications 69 (2016) 134–151.

[22] F. Paci, C. Fernandez-Gago, F. Moyano, Detecting insider threats: a trust-aware framework, in: IEEE (ed.), 8th International Conference on Availability, Reliability and Security (ARES), 2013, pp. 121–130.

[23] H. Psaier, L. Juszczyk, F. Skopik, D. Schall, S. Dustdar, Runtime Behavior Monitoring and Self-Adaptation in Service-Oriented Systems, 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO) 0 (2013) 164–173.

[24] Y. Qin, Q. Z. Sheng, N. J. Falkner, S. Dustdar, H. Wang, A. V. Vasilakos, When things matter: A survey on data-centric internet of things, Journal of Network and Computer Applications 64 (2016) 137 – 153.

[25] Y. B. Saied, A. Olivereau, D. Zeghlache., Trust management system design for the internet of things: A context-aware and multi-service approach, Computers & Security 39 (2013) 351–365.

[26] Q. Vu, S. Hassas, F. Armetta, B. Gaudou, R. Canal, Combining trust and self-organization for robust maintaining of information coherence in disturbed mas, in: Fifth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), IEEE, 2011, pp. 178–187.

[27] Z. Yan, C. Prehofer, Autonomic Trust Management for a Component-Based Software System, Dependable and Secure Computing, IEEE Transactions on 8 (6) (2011) 810–823.