Using Reference Architecture for Design and Evaluation of Web of Things Systems [1]

A Case of Smart Home Domain

Muhammad Aufeef Chauhan[2,a,*,†] and Muhammad Ali Babar[*,‡]
* IT University of Copenhagen, Centre for Research in Engineering Software Technologies, Software and Systems Section, Copenhagen, Denmark
† Technical University of Denmark, Department of Applied Mathematics and Computer Science, Lyngby, Denmark
‡ The University of Adelaide, Centre for Research in Engineering Software Technologies, School of Computer Science, Adelaide, Australia
[a] Corresponding: muac@itu.dk

Abstract

Web of Things (WoT) provides abstraction that simplifies the creation of Internet of Things (IoT) systems. IoT systems are designed to support a number of ubiquitous devices and decision management sub-systems. The devices and sub-systems can be a part of safety critical operations as well as smart management of multiple actuators that control the smart home devices. The devices and sub-systems need to comply to standardized business and quality requirements of a specific IoT domain. Designing sub-systems and actuators for the individual devices independently can result in lack of standardization and negatively impact the overall quality of the IoT system. Standardisation of the IoT applications constituting IoT system can be facilitated by providing a standardisation at the architecture level. As using Reference Architectures (RA) is a well established approach to achieve architectural standardisation, using the RA for designing IoT systems can facilitate standardisation of the architecture of individual sub-systems in an IoT system. The aim of the research presented this chapter is to provide insight to the process of using the RA for analysis, design, evaluation and evolution of the IoT systems. We present a software process-based approach to use the RA for development of IoT system. We use a case study-based research approach to analyse application of the process for design, evaluation and evolution of the IoT system for smart-home domain. The applications of the presented approach is analysed with reference to security and energy management in the smart homes. The results of the case studies show that (i) the IoT RA can be used for the initial design to incorporate the standardized business and quality requirements, (ii) the elements of the concrete sub-system architectures can be included in the IoT RA for its evolution with respect to the emerging requirements and (iii) open discussion by including all the potential stakeholders to determine key business and quality requirements of an IoT system can play an important rule for the evaluation of the concrete IoT sub-systems as well as evolution of the IoT RA. We foresee

[1]This chapter describes a method for software architecture design, analysis, evaluation and evolution of individual Internet of Things (IoT) subsystems and how individual IoT subsystems can be used for design of Web of Things (WoT) systems.

[2]Muhammad Aufeef Chauhan is a Postdoctoral researcher at IT University of Copenhagen and Technical University of Denmark.

that the presented research can be used for the analysis, design, evaluation and evolution of the IoT systems in distributed arrangements.

Chapter points

This chapter presents following key contributions:

- An methodology for analysis, design, evaluation and evolution of Internet of Things (IoT) subsystems and Web of Things (WoT) system using a specific IoT Reference Architecture (RA).
- An approach for analysis and evaluation of the key business requirements and quality constraints of the IoT systems.
- A comprehensive case study for the application of the presented approach for smart homes.

1. Introduction

Web of Things (WoT) aims to provide a platform for smart things also referred as Internet of Things (IoT) including sensors and actuator networks, embedded devices, digitally enhanced objects and decision support systems [1]. Whilst the IoT researh primarily focuses on providing connectivity in variety of networking environments, WoT focuses on application layer [1]. Though the software constituting WoT for different IoT devices and device management systems are designed and developed by numerous independent organizations, these need to operate as part of a specific WoT ecosystem (domain). Developing the software for the devices and corresponding server side systems in a manner that these can comply with essential business requirements and quality constraints is not trivial. The requirements can be related to general quality of the software system such as security of data [2] and services [3], as well as infrastructure (on which the data and services are hosted) specific quality requirements such as availability, scalability, Service Level Agreement (SLA) compliance and interoperability [4]. As the different services and components constituting IoT ecosystem can be developed independently, there is a need to have a standardization mechanism that can facilitate the analysis, design, evaluation and evolution of WoT architectures. Reference Architecture (RAs) is used to provide either a standardisation of the concrete architectures or to propose a preliminary architectural proposition that can be used to design concrete architectures [5]. The RAs consists of not only different elements of the architecture in terms of architecturally significant requirements, architecture components and architecture views [6] but also the process for designing, evaluating and instantiating a RA into concrete architectures [5]. As RA encompasses a complete life cycle of instantiation of a specific type of RA into multiple concrete architectures, choosing a RA-centric approach for architectural standardisation of sub-system architectures in WoT can facilitate achievement of standardisation and quality.

In this chapter, we present a RA-centric process-based approach for analysis, design and evaluation of the multiple concrete architectures driven by a single RA. We also provide insight on how the detailed design and instantiation of the concrete architectures can lead to the evolution of the RA. We have considered smart home IoT systems as a specific case of WoT for the application of the proposed approach. In particular we have addressed following research objectives.

- We have presented a RA-centric process-based approach that focuses on using a unified RA for design of the concrete architectures of all the encompassing IoT subsystems for a given WoT domain. The presented approach also focus on identifying variability and evolvability points in the RA that can change and evolve with the evolution of the concrete architectures.
- We have analysed the benefits of participation of stakeholders' involved in concrete architecture design of WoT sub-systems to streamline core business and quality requirements of the domain.
- We have demonstrated application of the proposed approach for security and energy management in smart home IoT subsystems.

The chapter is organized as follows. Section 2 provides details on the proposed RA-centred architecture analysis, design, evaluation and evolution approach. Section 3 provides details on the case study of application of the approach for designing ten IoT subsystems that lead to design of a WoT system for smart homes. Section 4 describes related work. Section 5 concludes the paper and provides insight to the experiences and lessons learned.

2. Architecture Design Considerations for Web of Things Systems

In the Web of Things (WoT) concepts, smart things and their associated services are fully integrated by leveraging technologies and patterns that are used to develop traditional web-based systems [1]. As a result the things and services constituting WoT has to be compliant with a core set of quality requirements. A compromise in the core quality requirements by any one of the thing or service forming WoT, can result in compromise on the quality of whole WoT system. Hence it is important to consider quality in the architecture design of each of the subsystems. In this section, we first describe key architecture quality concerns for WoT with reference to IoT. We then describe design guidelines for architecture of the individual things and services using the RA, strategies for evaluation of the architectures and how concrete architecture design can be used for the evolution of the RA. Figure 0.1 shows the pictorial representation of the analysis, design and evolution process.

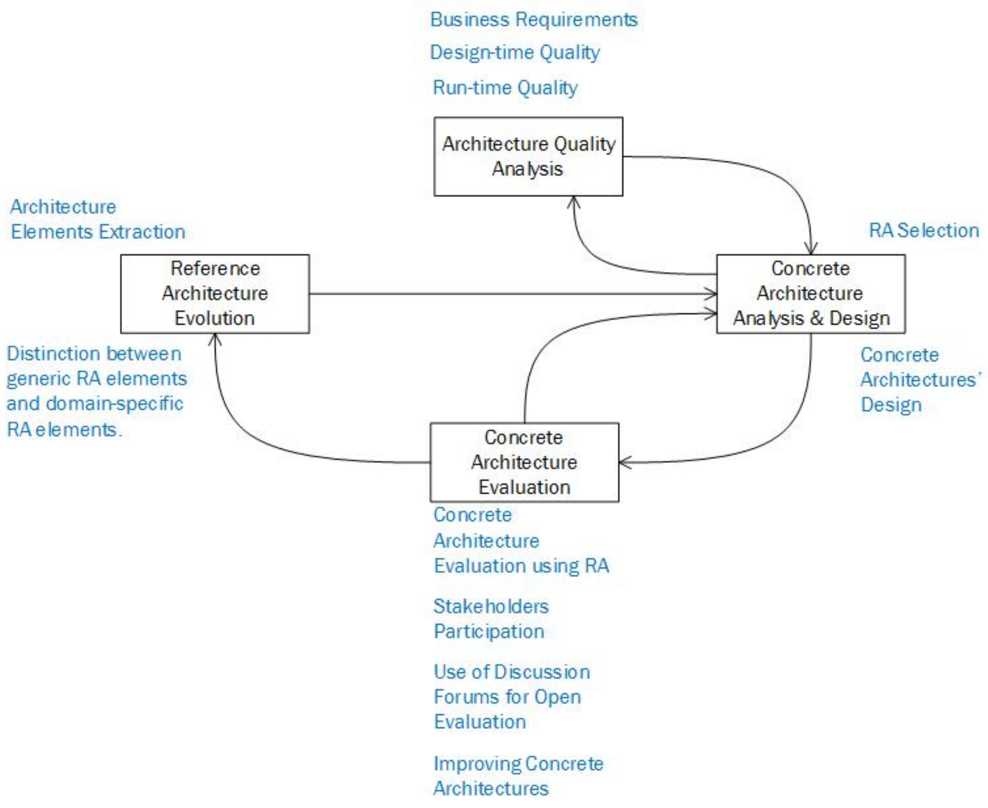
Business Requirements
Design-time Quality
Run-time Quality
Architecture Quality Analysis
Architecture Elements Extraction
RA Selection
Reference Architecture Evolution
Concrete Architecture Analysis & Design
Distinction between generic RA elements and domain-specific RA elements.
Concrete Architectures' Design
Concrete Architecture Evaluation
Concrete Architecture Evaluation using RA
Stakeholders Participation
Use of Discussion Forums for Open Evaluation
Improving Concrete Architectures

Figure 0.1 Architecture Design and Evaluation Process

2.1. Key Architecture Quality Considerations

As things and services constituting a WoT concept operate compositely, to achieve the quality in the WoT concept, all the things and services need to adhere to a core set of quality requirements. Absence of quality in software of any of the participating things or service can negatively impact the quality of the whole system. Hence it is important to select a minimum set of quality attributes that can be eligible for consideration while designing WoT concepts. The quality attributes can be related to design-time and run-time quality of the things and services, quality of deployment infrastructure and quality of communication infrastructure. In this section, we provide an overview of the key quality attributes for WoT with reference to IoT.

2.1.1. Security

Security is a key quality attribute of every distributed system [4] and guarantees that the things and services are secured from un-authorised access, data and messages are secured from being corrupted while transmission, and are persisted according to required security parameters of a specific system. The security quality attribute can be broken down into three main sub-attributes: (i) *Access Control*, (ii) *Communication Security* and (iii) *Persistence Security*. *Access Control* guarantees that things and services are accessible only by authenticated client services to protect these from unauthorized access and only the allowed level of information is revealed to the clients based upon their access level in the authorization hierarchy. *Communication Security* requires that the data and messages and encrypted while transmission so that these are protected from stealing. *Persistence Security* requires that the data is stored in an encrypted manner both by the things and by the services, and encryption keys are securely handled so that even if some of the things or services are confiscated, the data is protected from malicious use.

2.1.2. Availability

The WoT concepts primarily focus on application layer [1] and require to guarantee the availability of all the composite things and services. Hence, a WoT architecture has to ensure smooth operation of the services even if some of (i) the things and services are out of operation, (ii) the communication channels are broken or (iii) these are attacked by an insider (e.g., by any of the thing or service) or by an outside (e.g., a hacking attack or a denial of service attack [7]). Availability is also important from deployment perspective. If the things and services in IoT cannot perform the desired operation, then new things are to be activated and new service instances are to be deployed on server side.

2.1.3. Scalability and Elasticity

Scalability and elasticity of a WoT system is required to complement the availability. The number of active devices in an IoT system can vary and the data that is generated by the devices can increase or decrease. As a result the corresponding server side services need to be scaled up or down. There can be a large number of things and services in a WoT system as well as the users that interact with these. A highly scalable and elastic infrastructure such as Infrastructure as a Service (IaaS) cloud can be viable option to achieve scalability and elasticity [4] in WoT systems.

2.1.4. Reliability

Guaranteeing reliability of the things and services in a WoT system can play a vital role in adoption of the system. Reliability requires that a specific configuration of the things and services in a WoT system can be able to complete the desired operations and the produced results are correct. Therefore, a WoT system should allocate services to the things in an IoT mesh according to the requirements of the operations and should have redundancy mechanisms in place to guarantee variability of the produced results.

Fault tolerance approaches such as Byzantine Fault Tolerance model can be used for reliability verification [8].

2.1.5. Multi-tenancy

Multi-tenancy is an important characteristic of the system and requires providing isolation among the data and services belonging to different tenant (user groups). Multi-tenancy is handled in different manners including: (i) Access control based on the tenants’ roles that they can perform in a system and the type of data they can access to [9][3], (ii) Sharing services among the tenants having similar quality constraints [10], (iii) Scheduling tasks on the resources that can achieve the desired quality [11]. (iv) Periodic monitoring of the services for their suitability for the tenants [12].

2.1.6. Interoperability

Interoperability quality enables a software systems to work with other software systems [13]. Interoperability of things and services in WoT is important to facilitate their composition and provisioning. Interoperability is also critical from infrastructure perspective that is used to host the services in a WoT system. For example, if heterogeneous IaaS cloud infrastructures are used to host the services, their capability to host the services constituting WoT is important so that an appropriate IaaS cloud, which matches the security and reliability constraints, can be chosen. A number of architectural constructs can be adopted in the WoT system architecture as well to support interoperability. For example, proxies and services’ facades can hide the internal detail of how the services are deployed and migrated among IaaS clouds during their lifecycle [14]. Autonomous conversion of information associated with service semantics into service syntax can facilitate services’ interoperability [15]. Adopting a layered architecture approach can be useful to compartmentalise the services and providing interoperability among the services belonging to similar layers [16]. A strategy to delegate tasks to the optimal services configuration and dynamically allocating hosting IaaS cloud resources can ease the process of achieving services’ interoperability [17].

2.2. Concrete Architecture Design

An architecture design process guides the analysis and design of the software architectures [18]. The architecture analysis and design process for WoT systems needs to consider all possible things and services for which a particular WoT system is envisioned. As a WoT system aims to provide re-usability and adaptability of the things and services, the design process needs to focus on providing a standardization. An starting point to have a standardised system design is to analyse design of sub-systems by using a common architecture template, which is known as a Reference Architecture (RA) [5]. There are two main stages for WoT architecture deaign: (i) selecting an appropriate RA that is compliant with desired quaity requirements and (ii) using the selected RA for analysis and design of the concrete architecture.

2.2.1. Selecting an Appropriate Reference Architecture

While selecting the RA that is to be used as a baseline for analysis and design of the concrete WoT architecture, and a tool kit for standardization, first step is to analyse the essential quality requirements of the domain. Once the essential quality requirements are determined, all of the candidate RAs should be analysed for (i) whether a RA under consideration support the essential quality requirements or (ii) whether the RA supports addition of new components corresponding to a specific quality attribute without impacting the overall quality of the RA if there is no direct support for a specific quality attribute. A reference architecture providing a closest match with respect to the above mentioned conditions can be selected for the analysis and design of the WoT system architecture.

2.2.2. Using the selected RA for Analysis and Design of Concrete WoT Architecture

Once a RA is selected, it can guide the analysis and design of the concrete architecture of WoT system as well as its constituting IoT sub-systems. First step after selecting a reference architecture is to analyse it for its adoption for the given IoT domain with respect to the core business and quality requirements. The analysis includes identification of the sub-systems and components that can be incorporated in the concrete architecture. There can be three cases for the analysis: (i) Identification of the RA elements (sub-systems and components) that can be incorporated in the concrete architecture without tailoring and modification. (ii) Identification of the RA components that can be incorporated in the concrete architectures but require tailoring. (iii) Identification of the elements that are not presented in the RA but are needed because of the IoT domain and critical quality requirements.

Based upon the analysis, individual IoT systems can be designed using the IoT RA. Then the IoT RA along with the additional elements that are incorporated in each of the individual IoT system are used for the design of the concrete WoT architecture that can facilitate interaction of things and services across multiple IoT subsystems. While designing additional elements (which are not presented in the RA) in the concrete architecture, existing design patterns [19] and architecture styles as well as architecture patterns can be used [20].

2.3. Concrete Architecture Evaluations

Once the WoT architecture and IoT sub-system architectures are designed, next step is to evaluate the architecture for business, functional and quality requirements of the IoT domain. Software architecture evaluation guarantees that the system design is compliant to the desired quality. A number of software architecture evaluation methods are reported including Architecture Tradeoff Analysis Method (ATAM) [21], Software Architecture Analysis Method (SAAM) [22] and Quality-driven Architecture Design and Analysis Method (QADA) [23]. ATAM, SAAM and QADA can be used to perform specific architecture evaluation activity e.g., identifying risks and non-risks of architecture design decisions, identifying sensitivity and trade-off points, performing trade-off analysis for conflicting design decisions and quality driven analysis of the architecture for its quality completeness [21][22][23]. However, for architecture evaluation of WoT and constituting IoT systems, a specialized meta-level architecture evaluation approach is required.

We propose the use of the RA as a baseline for evaluation and involving stakeholders in the architecture evaluation process. The architecture evaluation sessions needs to be organized in such a manner that architecture is evaluated by not only the stakeholders of the IoT subsystem that is being evaluated but also by the stakeholders of other IoT subsystems. Similarly, the architecture of WoT system should be evaluated by representative stakeholders of all the IoT subsystems. Tailored ATAM, SAAM and QADA methods can be used to perform specific evaluation activities. The WoT system's architecture and IoT subsystems' architectures should be distributed among the participants of the sessions well in advance. The feedbacks from the evaluation sessions need to be analysed by the committee of the core architects to make sure that the design decisions made during multiple architecture evaluation sessions are harmonious. We also propose the use of discussion forums to share design decisions across evaluation teams to mitigate the risk of conflicting evaluation decisions.

2.4. Reference Architecture Evolution

The RA used as a baseline for the concrete architecture design needs to be evolved to cater emerging requirements of a specific IoT domain. Albeit a clear distinction should be made between generic RA elements and domain specific RA elements. The additional elements (as described in Section 2.2.2) from concrete WoT architecture and each of the concrete IoT architectures needs to be extracted and added into the RA. The RA evolution facilitate the design of concrete IoT sub-systems constituting a WoT system and facilitate its evaluation as the RA already includes findings on the evaluation from the previous design and evaluation activities of the RA's concrete instances. However, only the elements associated with business or quality requirements should be included in the RA. Elements related to functional requirements should not be included in the RA. (to be updated: quality calculator for branches of the RA)

3. A Case Study for application of the Approach in Smart-home Domain

We have used the architecture design considerations presented in Section 2 for design of concrete WoT and IoT architectures for the smart home domain. A case study was conducted with participation of thirty master level course students and 3 course instructors. The students were divided into group of three each and were given a task to design a concrete architecture for a particular aspect of smart home system using the IoT reference architectrue. The students were given a brief description of the smart home domain and were asked to choose a specific subsystem as per their preference and knowledge with the domain. Whereas, the instructors participated in design and development WoT architecture that can facilitate interaction of multiple smart home subsystems via web. All the students have a bachelor degree in information technology related discipline and have industrial experience with design and development of web-based system. Two of the course instructors have PhD in software engineering and one course instructor have a master degree in software engineering, all with extensive experience with architecture design and development of distributed and web-based software systems.

3.1. Streamlining Business and Quality Requirements

First step of the concrete IoT architecture design was to streamline core business and quality requirements of a specific IoT subsystem. As the students group were given a choice to select an IoT subsystem for the concrete design of the architecture, they choose different aspect of IoT smart home. The groups has been assigned the number from 1 to 10 (e.g., Group 1). Table 0.1 shows the different subsystems chosen by each group. The subsystems can correspond to core business requirements of the smart home IoT system. As some groups select more than one aspects of smart home IoT system (e.g., Group 5 chooses management of different appliances in smart homes and safety management in case of an incident of fire), these are shown corresponding to multiple subsystems in Table 0.1.

Streamlining quality requirements in individual IoT systems is important so that a collective set of quality requirements for WoT architecture, to facilitate individual IoT subsystems management, can be determined. A two faceted approach was adopted to determine a set essential quality requirements. In a first stage, the individual groups identified the quality requirements specific to the IoT subsystems they considered for architecture design. In second stage, a public discussion forum was used to discuss the quality requirements of the IoT subsystems that two and more of the group were considering for the design. Table 0.1 shows the finally agreed quality requirements for different IoT subsystems. It is clear from Table 0.1 that core quality requirements (Security, availability, scalability/elasticity, Reliability, Multi-tenancy and Interoper-

Table 0.1 IoT Subsystems and Quality Attributes

IoT Subsystems	Groups IDs	Quality Attributes
Devices and Appliances Management	Group 1, Group 4, Group 5, Group 6, Group 7, Group 8, Group 9, Group 10	Performance, Usability, Reliability, Security, Modifiability, Interoperability, Multi-tenancy, Availability, Scalability, Elasticity
Fire and Safety Management	Group 2, Group 3, Group 5	Availability, Reliability, Scalability, Performance, Failure Recovery, Interoperability, Modifiability, Security
Home Access and Security	Group 3, Group 6, Group 7, Group 8, Group 9	Interoperability, Availability, Modifiability, Performance, Security, Reliability, Usability
Monitoring and Surveillance	Group 7, Group 10	Security, Reliability, Performance, Usability, Elasticity, Interoperability, Multi-tenancy

ability) presented in Section 2.1 is considered for all the IoT subsystems.

3.2. Selecting and Using IoT Reference Architecture for Preliminary Architecture Design

The next step is to select an IoT Reference Architecture (RA) that provides support and flexibility to design concrete IoT architectures by incorporating desired quality attributes (characteristics) in the RA and by tailoring the RA in terms of adding additional components and excluding undesired components from a concrete IoT architecture. Our analysis of the RAs for IoT revealed that Bauer et al. [24] have presented the most comprehensive RA for IoT things domain. We selected their RA as a baseline for design and analysis of the concrete architectures for IoT subsystems and subsequently WoT architecture to manage individual IoT subsystems. Figure 0.2 shows the IoT RA presented by [24]. The project groups used the selected RA as a baseline to design the concrete IoT architectures.

3.2.1. Using IoT RA Architecture for Design of Concrete IoT Systems for Smart Homes

After selecting the IoT RA, the students groups were asked to use the RA as a baseline for the design of the conrete IoT architectures for the specific business and quality requirements (as elaborated in Section 3.1). Following design strategies were dopted

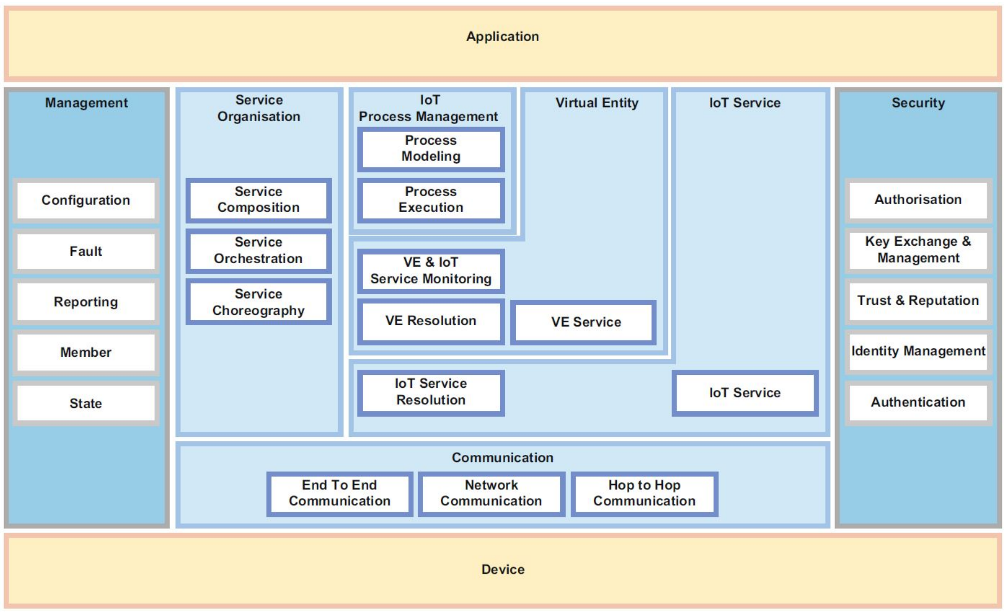

Figure 0.2 Internet of Things (IoT) Reference Architecture (RA) presented in [24]

for designing the conrete architectures.

- IoT RA layers were analysed to identify main layers of the IoT system architectures.
- The components of the IoT RA that could be directly adopted in the concrete IoT architectures were analysed and adopted. For example, *Authentication* and *Authorisation* components from the *Security* layer can be adopted directly into the concrete architecutres.
- The components that required tailoring or specialisations were modified according to the requirements of the target concrete architectures. For example, *Communication* layer of the IoT RA have identified the components for end-to-end communication, network communication and hop-to-hop communication. These components needed to have a specializations for the specific type of communication protocols that were to be used by the concrete IoT architectures.
- Additional components were added for the layers of the IoT RA that were only providing a skeleton of the architecture. For example, components for the specific services were added into *IoT Service* layers and components for managing and deploying services were added into *Virtual Entity* layer.
- To incorporate the desired quality attributes in the concrete IoT architectures, multiple architectures styles and patterns [20] as well as design patterns [19] were used.

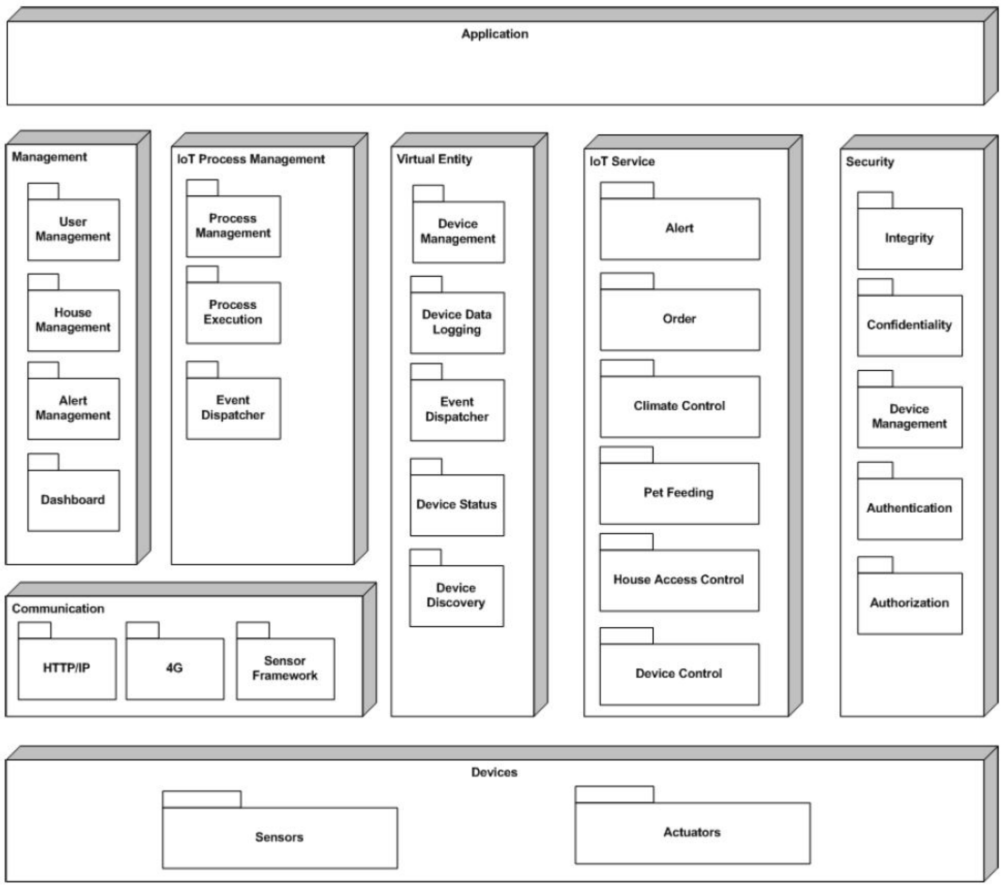

Figure 0.3 IoT System Architecture for Devices Management and Home Security

Figure 0.3 shows a representative concrete IoT system architecture for management of the devices in the smart homes designed to achieve business and quality requirements related to devices management in the smart homes. As shown in the figure, the layers and components from the IoT RA have been tailored and modified. *Devices* layer provides interfaces to connect to and receive information from different types of the sensors. This layer also provides interfaces through which devices and actuators in smart homes can be controlled. *Communication* layer includes components for *Http and TCP/IP*, *Mobile communication* (such as 4G) and *Sensor Framework* (to accommodate devices specific communication protocols). Other layers of the architecture interacts with the devices via *Communication* layer.

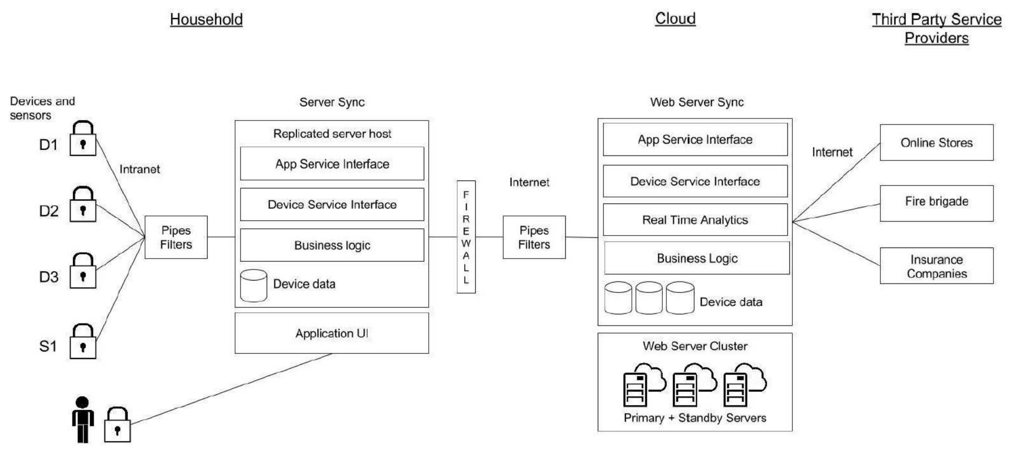

Figure 0.4 IoT System Distribution and Interaction

Security layer includes components for *Authentication*, *Authorization*, *Integrity* verification, *Confidentiality* handling of the data and secured collection of data from the devices (via *Device Management* component). *IoT Service* layer include components corresponding to specific devices in the smart homes. For example, *House Access Control* component manages access to the home by the householders, *Climate Control* components controls indoor temperature and *Pet Feeding* components put food and water into pet food utensils. *Virtual Entity* layer manages devices (turning the devices on and off), data and event logging, and recovery of system configuration if some of the devices malfunction. *IoT Process Management* layer handles definition and execution of the rules and processes for management of the smart homes. *Management* layer provides Graphical User Interfaces (GUIs) and tools for through which users can manage and control the devices.

Figure 0.4 shows deployment configuration of IoT system in distributed arrangement in which the components are hosted on in-house infrastructure as well as on external hosting platforms to adequately handle system failure scenarios. The devices and sensors can connect to only the in-house components. The communication between internally hosted and externally hosted components it filtered through a firewall to protect the system from undesired access. Connection to the external systems e.g., online stored for ordering of the items is managed by externally deployed components to minimize internal system exposure to the external world.

A number of architecture and design patterns were used to achieve quality attributes. Table 0.2 lists commonly used architecture and design patterns used in different IoT subsystems.

Table 0.2 Design Tactics Used to Achieve the Quality Attributes

Quality Attributes	Design Tactics
Performance	Distributed Components, Pipes and Filters, Prioritized Message Queues and Data Indexes
Security	Secured Proxies, Authenticaion, Authorization, and Session Tokens
Modifiability	Adapter, Facade, Loosely-coupled Layers/Components, Layered Architecture, Components/Services-based Architecture and Model View Controller (MVC) Pattern
Availability	Components' Redundancy and In-house hosting of Critical System Components
Reliability	Monitor Analyse Plan Execute (MAPE) Pattern, Redundancy of Components and Data and Hybrid Cloud Infrastructure for hosting Systems
Scalability	Data Input Queues, Hybrid Deployments and Load Balancing
Auditing	Event Logs,
Multi-tenancy	Tenant-specific Publisher/Subscriber Architecture, and Tenant Profiling and Management
Interoperability	Components' Face and Broker Pattern

3.2.2. Web of Things Architecture to manage constituting IoT Subsystems

Contrary to concrete IoT smart home architectures, which focus on providing different types of smart home services, connectivity with the respective devices and sensors; Web of Things (WoT) architecture focuses on application layer [1]. Figure 0.5 shows a high level view of the WoT system architecture designed to support smart home IoT subsystems. The WoT system architecture is designed using concrete IoT subsystem architectures and abstracting the application layer. The WoT system architecture focuses on provisioning of the IoT subsystems' components and services on the suitable hosting infrastructure, provide interfaces for interaction of the subsystems with external systems, managing Quality of Service (QoS) [4] parameters that are used for provisioning and composing IoT subsystems to provision a system configuration that can satisfy desired business, functional and quality requirements. The concrete IoT subsystems can be either hosted using WoT system architecture or can be managed

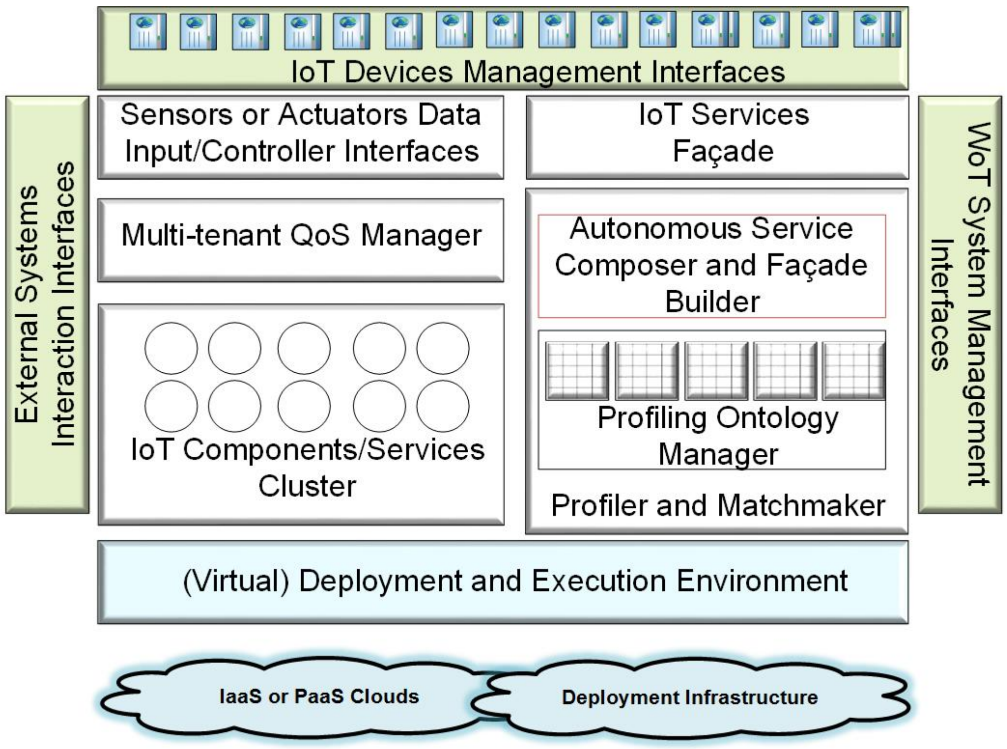
IoT Devices Management Interfaces
External Systems Interaction Interfaces
Sensors or Actuators Data Input/Controller Interfaces
IoT Services Façade
Multi-tenant QoS Manager
Autonomous Service Composer and Façade Builder
Profiling Ontology Manager
Profiler and Matchmaker
IoT Components/Services Cluster
WoT System Management Interfaces
(Virtual) Deployment and Execution Environment
IaaS or PaaS Clouds
Deployment Infrastructure

Figure 0.5 Web of Things System Architecture

using the WoT system.

The WoT architecture presented in Figure 0.5 has been designed following separation of concerns and layered architecture approaches [20]. The architecture has three interface layers named as *IoT Devices Management Interfaces*, *External System Interaction Interfaces* and *WoT System Management Interfaces*. *IoT Device Management Interfaces* are used to receive input from the sensors of an IoT system and controlling devices and actuators. For example in IoT subsystem used for temperature control in a smart home, the hosted components and services can receive temperature readings from temperature sensors and can turn on heating or cooling system accordingly. *External System Interaction Interfaces* are used to connect hosted IoT subsystem services with external systems. For example in an IoT smart home subsystem for security management in a smart home, the subsystem interacts with emergency police services and notify home owners in case of a break-in or burglary. *WoT System Management Interfaces* are used to define tenants, their desired QoS parameters and specify Service Level Agreements (SLAs) [4] of the IoT subsystem.

The core WoT system components are classfied into two categories: (i) the components that are responsible for IoT subsystems selection and composition for desired QoS parameters and (ii) the components that are responsible for the provisioning of the subsystems' components/services on dedicated or virtualized infrastructure. *Profiler and Matchmaker* component maintains profiles of all the components/services of the hosted IoT subsystems using *Profiling Ontology Manager*, autonomously compose components/services and build on-the-fly interfaces using *Autonomous Service Composer and Facade Builder*. Ontology meta-models and service selection ontologies [25] can be used for profiling of the components/services. The interfaces are exposed via *IoT Services Facade* and *Sensors or Actuators Data Input/Controller Interfaces* itermediate layers. *Multi-tenant QoS Manager* is used to maintain desired QoS parameter for each specific tenant so that the services can be composed and provisioned accordingly. *Deployment and Execution Manager* is used to deploy the IoT subsystem components and either dedicated or virtualized infrastructure (e.g., IaaS or PaaS clouds [4]). *Deployment and Execution Manager* can be tailored using the RA for services selection and deployment on the cloud [25]. For cases in which some an IoT system is partially deployed on in-premise infrastructure and partially on off-premise infrastructure, the locally hosted and remotely hosted components can interact via *IoT Services Face*.

3.3. Evaluation of IoT Subsystems Architectures

Evaluation of the software architectures plays a critical role to verify quality of a software system. A number of approached have been proposed to carry out specific software architecture evaluation activities including but not limited to ATAM [21], SAAM [22] and QADA [23] as discussed in Section 2.3. As the aim of the IoT subsystem architecture design activity was to carry out design of the IoT subsystems corresponding to different high-level business requirements, the architecture evaluation activities had to verify compliance of the IoT subsystems to agreed upon standardized quality requirements (as discussed in Section 3.1) and design choices (as discussed in Section 3.2.1) that are not contradictory to each other. Hence the evaluation activities needs to be carried out in a way that the stakeholders involved in the design of differeent IoT subsystems can have maximum insight into the design of other system. The following subsections describe the details on the evaluation activities and how the IoT subsystems evaluation can lead to the evaluation of WoT system architecture as well as evolution of the IoT RA. An overview of the evaluation process is presented in Figure 0.6.

3.3.1. Organization of the Evaluation Sessions

Evaluation sessions for the evaluation of the IoT subsystems' architectures were organized in a way that stakeholders involved in design of IoT architecture covering one type of business requirements, evaluate IoT architectue designed to satisfy another

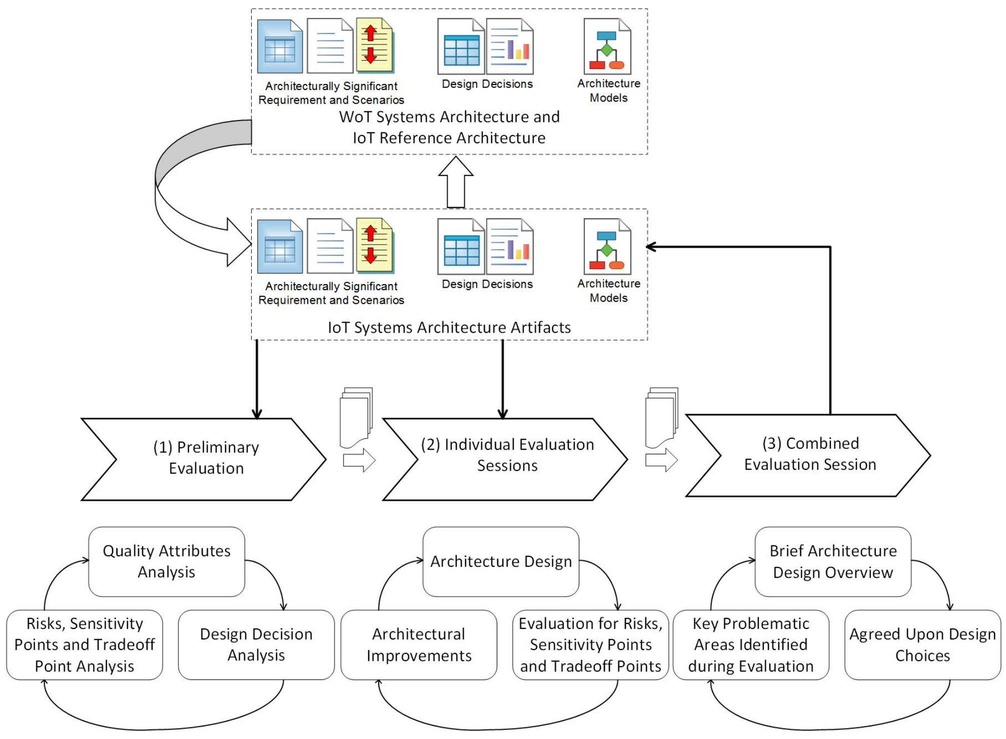
Architecturally Significant Requirement and Scenarios
Design Decisions
Architecture Models
WoT Systems Architecture and IoT Reference Architecture
Architecturally Significant Requirement and Scenarios
Design Decisions
Architecture Models
IoT Systems Architecture Artifacts
(1) Preliminary Evaluation
(2) Individual Evaluation Sessions
(3) Combined Evaluation Session
Quality Attributes Analysis
Risks, Sensitivity Points and Tradeoff Point Analysis
Design Decision Analysis
Architecture Design
Architectural Improvements
Evaluation for Risks, Sensitivity Points and Tradeoff Points
Brief Architecture Design Overview
Key Problematic Areas Identified during Evaluation
Agreed Upon Design Choices

Figure 0.6 Evaluation Process Overview

type of business requirements. For example, as presented in Table **??** Group 1 focused on IoT subsystem for management of devices and appliances in smart homes to have efficient utilization of electricity and water, Group 2 focused on IoT subsystem for safety management and fire extinguishing process in case of fire, and Group 3 focused on IoT subsystem for access control and security of the smart homes. Group 1 architecture was evaluated by Group 2, and Group 2 architecture was evaluated by Group 3 so that Group 2 can have insight to both of the device management and security subsystems. The architecture design documents were shared with the group who was to evaluate the architecture before the session. For example, Group 1's architecture documentation was shared with Group 2 and Group 2's architecture documentation was shared with Group 3.

3.3.2. Architecture Evaluation Activities

Architecture evaluation activities consisted of three stages. (i) Before the evaluation session, the groups prepared a short architecture evaluation questionaire on quality attributes considered in the architecture design, key architecture design decisions, strengiths and weeknesses of the design decisions, sensitivity and trade off points, and risks and non-risks in the architecture. (ii) During the evaluation sessions, the group whose architecture was evaluated presented the architecture while the group who was evaluating the architecture asked questions based upon their initial preparation as well as from integration perspective of their own architecture. During the architecture presentation design artefacts were analysed for evaluation of the IoT subsystem architectures as well some new artefacts specific to architecture evaluation such as architecture utility tree [21] were created to present architecture design decision corresponding to quality attributes were drawn. During the architecture evaluation sessions, sensitivity points, trade off points, architecture risks and architecture non risks were discussed. (iii) After the individual architecture evaluation sessions, a joint session was conducted in which each group briefly presented their IoT subsystem architecture, quality attributed those were considered in the architecture and feedback it received during the individual architecture evaluation session. Each group member also prepared a written report on the evaluation of the architecture that the group member has evaluated in the architecture evaluation session. The report was shared with teaching staff show were responsible for design and analysis of the WoT architecture. Table 0.3 shows commontly reported missing quality requirements, risky design decisions, sensitivity and trade-off points, and common improvements suggested in the IoT subsystem architectures.

3.3.3. Improvements in IoT Subsystem Architectures

All the groups improved IoT architectures of their respective systems based upon the changes suggested during architecture evaluation sessions. While improving the architectures, the findings from the individual evaluation sessions as well as joint follow up session were considered (as described in Section 3.3.2). A change log was maintained by the group to indicate (i) what changes had been made in the architecture, (ii) what was the source that triggered the change i.e., feedback from individual evaluation session or finding from joint session and (iii) trace to the relevant place of the architecture documentation where change had been made.

3.4. Evolution of Web of Things System and Internet of Things Reference Architecture

The improved architecture artefacts generated as a result of architecture evaluation of IoT subsystems can be used for improvements in the WoT system architecture and evolution of the IoT RA. Because architecture of WoT system was designed based on the IoT subsystems (as described in Section 2.2.2), the newly added components can be included in the WoT system and modified components can be changed. The concrete IoT subsystem architectures can also lead to the evolution of the selected IoT RA for a specific type of domain. A large-scale empirical study on challenges with designing and using RAs have revealed that unavailability of the details on design decisions

Table 0.3 Summary of the Points Discussed in the Evaluation Sessions

Dimension	Details
Risks	Using same communication protocol for inter-system and intra-system communication can make failure safety procedures difficult. Using broker pattern can result in selection of services not fully compliant with QoS constraints. Data storage on centralised persistance units is a risk for availability. Communication overhead of using Publisher/Subscriber pattern.
Sensitivity and Trade-off Points	Maintaining Facade for continuously evolving services. Using Singleton pattern instead of Message broker pattern for multi-tenant session management. Negative impact of layered pattern on performance. Trade-off between performance and modifiability by using either publisher scriber or layered pattern. Misuse of Pipes and Filter pattern where Broker pattern can be used. Trade-off is needed vs. Security and Performance
Commonly Missing Quality Attributes	Safety on Failure, Unavailability of Internet Connection
IoT RA Shortcomings	Architecture and design patterns for implementing different elements of the IoT RA are not mentioned.

and how to materialize the decisions using architecture styles and patterns can lead to problems while using the RA for concrete architecture design [26]. Same challenge was also reported by the teams who were designing IoT subsystems architectures using the IoT RA (as discussed in Section 3.3.2). Including architecture design decisions rationale and corresponding architecture and design patterns in the RA can faciliate its adoption. For example, including architecture and design patterns (presented in Table 0.2) in the IoT RA [24] (Table 0.2) can facilitate its adoption for smart home domain. However, variability points in the architecture should be considered while so to keep generic elements of the RA and domain specific elements of the RA distinguishable.

4. Related Work

Since the inception of the software architecture [27], a number of methods have been proposed for architecture design. Most prominent of these methods are Attribute Driven Design method [28], Siemens Four Views method, Rational Unified Process [6], Business Architecture Process and Organization and Architectural Separation of Concerns method [18]. All these methods has three main steps: architecture analysis, architecture synthesis and architecture evaluation [18]. Architecture analysis and synthesis activities include identifying architecturally significant requirements, developing architecture scenarios, designing architecture elements using architecture patterns and representing architecture using multiple views [6]. Generic architecture evaluation methods include Architecture Tradeoff Analaysis Methods (ATAM) [21], Software Architecture Analysis Method (SAAM) [22] and Quality-driven Architecture Design and Analysis method [23]. These methods focus on evaluating software architectures with respect to specific quality attributes by evaluating architecture strengths and weeknesses, sensiticity and tradeoff points, and completeness of a given architecture.

Other than generic architecture design methods, the methods for specific technological paradigms have also been proposed. The methods for Service Oriented Architectures (SOA) focus on transforming system elements in to software services, communication among the services, and exception handling and fault recovery strategies [29][30][31], The methods for cloud-based systems focus on satisfying Quality of Service (QoS) requirements (e.g., availability, security, safety), cloud specific quality requirements (e.g., multi-tenancy and cloud-interoperability) and selecting appropriate cloud resources or cloud-hosted services that satisfy Service Level Agreements [4][32].

The increased in the complexity of the software systems has raised the need to have methods for reusable architecture solutions. Hence, the RAs design and analysis methods are devised [5]. The RA focus on cepturing generic business and achitecture requirements and corresponding system representation. For example, a RA for cloud-based tools focus on tools selection, tools provisioning, providing semantic integration among the tools and raising awareness of the operations that are performed using the integrated tool [25]. However, using a RAs to design concrete architectures and evolving the RAs as the domain for which the RA is designed is a challenging undertaking [26]. The research presented in this chapter has attempted to bridge this gap by builing an analysis and design approach on top of the existing approached.

5. Conclusions and Lessons Learned

In this chapter, we suggested using a Reference Architecture (RA) centred approach for the detailed analysis, design, evaluation and evolution of the Internet of Things (IoT) and Web of Things (WoT) systems. After discussing the key architecture quality attributes for designing IoT and WoT systems (i.e., security, availability, scalability, reliability, multi-tenancy and interoperability), we argued that selection of the RA for concrete IoT system design should be based upon the RA's support for the desired quality attributed or its ability to support addition of architectural elements that can support the desired quality attributes. We have proposed to directly use the selected RA for design of the concrete IoT systems (i.e., using RA for dsigning IoT systems) and indirectly use the RA for design of concrete WoT system (using IoT systems as a baseline for designing WoT systems). Concrete IoT architectures can be evaluated using existing software architecture evaluation methods and the common components from the concrete architectures can be included into the RA to have a domain-specific representation of the RA (e.g., for smart home domain). We also have shown applicability of the proposed approach for design and evaluation of ten smart home IoT subsystems and corresponding WoT system.

We have made a number of observations while applying the approach for smart home domain as follows. If different subsystems of an IoT system are being designed by independent teams, it is important to ***streamline core quality requirements for all the IoT subsystems*** so that conflicting and contraditory architectural design decisions can be avoided. Using a core set of quality requirements in the subsystems and discussion forums that are accessible to all teams can be a good practice in this regard. Considering the generic nature of the RAs, these always need to be tailored for adoption in a specific domain. Hence, ***the selected RA should be flexible to accommodate components specific to functional requirements as well as additional quality requirements***. Contrary to using RA directly for analysis and design of WoT system for IoT subsystems, ***using concrete IoT architectures as a baseline for WoT architecture can provide a better opportunity for analysis*** of the common application components and their inclusion in the WoT system. ***Maintaining a log of all the design decisions, and architecture patterns and design patterns*** used in the concrete IoT systems' design can facilitate not only design of the IoT system but also evolution of the RA for future use. Organization of the individual evaluation sessions for different IoT subsystem in a way that ensure ***maximum participation of the teams*** involved in designing other systems can facilitate having common understanding of the domain. ***Maintaining variability points in the RA*** when it is evolved by incorporating domain specific elements from the concerete IoT subsystems can guarantee seperation of the generic RA elements and domain specific RA elements. We foresee that the presented approach can be used for design and evaluation of the complex WoT systems.

ACKNOWLEDGMENTS

We acknowledge contribution of Software Architecture 2015 course students at IT University of Copenhagen for their participation in design and evaluation of concrete smart homes IoT architectures.

REFERENCE

1. D. Guinard, V. Trifa, F. Mattern, E. Wilde, From the internet of things to the web of things: Resource-oriented architecture and best practices, in: Architecting the Internet of Things, Springer, 2011, pp. 97–129.
2. F. Yahya, V. Chang, R. J. Walters, G. B. Wills, Security challenges in cloud storages, in: Cloud

Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on, IEEE, 2014, pp. 1051–1056.
3. M. Almorsy, J. Grundy, A. S. Ibrahim, Tossma: A tenant-oriented saas security management architecture, in: Cloud computing (cloud), 2012 ieee 5th international conference on, IEEE, 2012, pp. 981–988.
4. M. A. Chauhan, M. A. Babar, B. Benatallah, Architecting cloud-enabled systems: a systematic survey of challenges and solutions, Software: Practice and Experience.
5. S. Angelov, P. Grefen, D. Greefhorst, A framework for analysis and design of software reference architectures, Information and Software Technology 54 (4) (2012) 417–431.
6. P. B. Kruchten, The 4+ 1 view model of architecture, IEEE software 12 (6) (1995) 42–50.
7. W. Huang, A. Ganjali, B. H. Kim, S. Oh, D. Lie, The state of public infrastructure-as-a-service cloud security, ACM Computing Surveys (CSUR) 47 (4) (2015) 68.
8. M. A. AlZain, B. Soh, E. Pardede, A byzantine fault tolerance model for a multi-cloud computing, in: Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on, IEEE, 2013, pp. 130–137.
9. J. B. Bernabe, J. M. M. Perez, J. M. A. Calero, F. J. G. Clemente, G. M. Perez, A. F. G. Skarmeta, Semantic-aware multi-tenancy authorization system for cloud architectures, Future Generation Computer Systems 32 (2014) 154–167.
10. H. Moens, E. Truyen, S. Walraven, W. Joosen, B. Dhoedt, F. De Turck, Cost-effective feature placement of customizable multi-tenant applications in the cloud, Journal of Network and Systems Management 22 (4) (2014) 517–558.
11. C. Fehling, F. Leymann, R. Mietzner, A framework for optimized distribution of tenants in cloud applications, in: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, IEEE, 2010, pp. 252–259.
12. F. R. Sousa, J. C. Machado, Towards elastic multi-tenant database replication with quality of service, in: Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, IEEE Computer Society, 2012, pp. 168–175.
13. E. M. Maximilien, A. Ranabahu, R. Engehausen, L. Anderson, Ibm altocumulus: a cross-cloud middleware and platform, in: Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, ACM, 2009, pp. 805–806.
14. L. S. Ribeiro, C. Viana-Ferreira, J. L. Oliveira, C. Costa, Xds-i outsourcing proxy: ensuring confidentiality while preserving interoperability, IEEE journal of biomedical and health informatics 18 (4) (2014) 1404–1412.
15. R. Rezaei, T. K. Chiew, S. P. Lee, Z. S. Aliee, A semantic interoperability framework for software as a service systems in cloud computing environments, Expert Systems with Applications 41 (13) (2014) 5751–5770.
16. Y. Charalabidis, S. Koussouris, A. Ramfos, A cloud infrastructure for collaborative digital public services, in: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, IEEE, 2011, pp. 340–347.
17. H. Flores, S. N. Srirama, Mobile cloud middleware, Journal of Systems and Software 92 (2014) 82–94.
18. C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, P. America, A general model of software architecture design derived from five industrial approaches, Journal of Systems and Software 80 (1) (2007) 106–126.
19. A. Shalloway, J. R. Trott, Design patterns explained: a new perspective on object-oriented design, Pearson Education, 2004.
20. F. Buschmann, K. Henney, D. C. Schmidt, Pattern-oriented software architecture, on patterns and pattern languages, Vol. 5, John wiley & sons, 2007.
21. R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, J. Carriere, The architecture tradeoff analysis method, in: Engineering of Complex Computer Systems, 1998. ICECCS'98. Proceedings. Fourth IEEE International Conference on, IEEE, 1998, pp. 68–78.
22. R. Kazman, L. Bass, M. Webb, G. Abowd, Saam: A method for analyzing the properties of software architectures, in: Proceedings of the 16th international conference on Software engineering, IEEE

Computer Society Press, 1994, pp. 81–90.
23. M. Matinlassi, E. Niemelä, L. Dobrica, Quality-driven architecture design and quality analysis method, A revolutionary initiation approach to a product line architecture, VTT Technical Research Centre of Finland, Espoo.
24. M. Bauer, M. Boussard, N. Bui, J. De Loof, C. Magerkurth, S. Meissner, A. Nettsträter, J. Stefa, M. Thoma, J. W. Walewski, Iot reference architecture, in: Enabling Things to Talk, Springer, 2013, pp. 163–211.
25. M. A. Chauhan, M. A. Babar, Q. Z. Sheng, A reference architecture for a cloud-based tools as a service workspace, in: Services Computing (SCC), 2015 IEEE International Conference on, IEEE, 2015, pp. 475–482.
26. S. Angelov, J. Trienekens, R. Kusters, Software reference architectures-exploring their usage and design in practice, in: European Conference on Software Architecture, Springer, 2013, pp. 17–24.
27. I. Gorton, Essential software architecture, Springer Science & Business Media, 2006.
28. L. Bass, M. Klein, F. Bachmann, Quality attribute design primitives and the attribute driven design method, in: International Workshop on Software Product-Family Engineering, Springer, 2001, pp. 169–186.
29. M. Razavian, P. Lago, A frame of reference for soa migration, in: European Conference on a Service-Based Internet, Springer, 2010, pp. 150–162.
30. M. Razavian, P. Lago, Towards a conceptual framework for legacy to soa migration, in: Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops, Springer, 2010, pp. 445–455.
31. G. Lewis, E. Morris, L. O'Brien, D. Smith, L. Wrage, Smart: The service-oriented migration and reuse technique (no. cmu/sei-2005-tn-029), Pittsburgh: Software Engineering Institute.
32. M. A. Chauhan, M. A. Babar, Towards process support for migrating applications to cloud computing, in: Cloud and Service Computing (CSC), 2012 International Conference on, IEEE, 2012, pp. 80–87.