Overview of the IoT Architectural Reference Model & Lessons learnt

Dr. François CARREZ
Institute for Communication Systems
5G Innovation Centre
Structure of presentation

1. IoT Architectural Reference Model
2. IoT Reference Model
3. Guideline & Process
4. Lesson Learnt
5. Home Work
1. IoT Architectural Reference Model
2. IoT Reference Model
3. Guideline & Process
4. Lesson Learnt
5. Home Work
IoT Architectural Reference Model

Overview

• Architectural Reference Model
  – “Meta-level”
  – How to specify an Architecture?
  – What is the terminology?
  – What are the views and perspectives?
  – Design choices & Guidelines

• Architecture
  – Specific instantiation following structure and guidelines of ARM
  – Specific components and interactions

• System
  – Concrete implementation according to Architecture
IoT Architectural reference Model

Genesis

IoT Architectural Reference Model

- SOTA
- Guidelines
- ARM-derived Concrete IoT Architectures

Existing architectures & solutions

Organisation

Implementation

IoT Reference Model

IoT Reference Architecture

Guidelines

ARM-derived Concrete IoT Architectures

Monday, 11 April 16
IoT Architectural Reference Model

What is it made of?

• **IoT Reference Model** to promote common understanding
  – High abstraction level
  – Describes the aspects of the IoT domain that do not change
  – Enables a general discourse on the IoT domain
  – Provides a Domain, Information, Functional, Communication and Security models

• **IoT Reference Architecture** to describe essential building blocks and identify design choices
  – Based on IoT Reference Model
  – Reference for building compliant IoT architectures
  – Provides views and perspectives on different architectural aspects that are of concern to stakeholders

• **Guidelines** to help in developing an architecture for a specific system based on the IoT Reference Architecture
  – Provide different types of guidance: process, usage per model and view, examples etc…
1. IoT Architectural Reference Model

2. IoT Reference Model

3. Guideline & Process

4. Lesson Learnt

5. Home Work
IoT Reference Model

IoT Domain Model (simplified version)

- High level description of the IoT Domain
- Set-up a common vocabulary/grounding
- Service-oriented approach
- Binding between VE’s and Services is dynamic
- Services offer one single end-point (e.g. REST) to access resources (resources stay in the background)
- VE-centric Services can also be used to expose VE’s attributes
• The Information Model models **Domain Model concepts** that are to be explicitly represented and manipulated in the digital world

• In addition the Information Model explicitly models **relations between these concepts**

• The Information Model is a **meta model** that provides a structure for the information

• This structure provides the basis for defining the **functional interfaces**
1. IoT Architectural Reference Model
2. IoT Reference Model
3. Guideline & Process
4. Lesson Learnt
5. Home Work
Guideline and Process

Objective

• The IoT ARM document is huge and provide comprehensive content about many aspects in developing an architecture, however the RM and RA alone do not answer the following questions:
  ✓ What are the activities?
  ✓ In which order should we undertake them?
  ✓ What is their outcomes
  ✓ Are they all described in the ARM?
  ✓ Are there more views, not listed in the ARM, to be considered as well?
  ✓ How do the activities link together in order to provide a consistent IoT System Architecture?
  ✓ How to handle requirements and when?

• Those simple and pragmatic questions are answered by the Guideline Chapter of the IoT ARM, and illustrated through concrete examples
Guidelines & Process

Overview

- Business goals
- Create Physical-Entity View
- Create IoT Context View
- Derive other views
- Requirement process
Guidelines & Process

IoT Physical Entity and Context Views

• Physical-Entity View:
  • Identifies the Objects (or Entities) of interest
  • What are their properties? which action can be performed upon them?
  • What are the supporting IoT Devices?
• Context View: focuses on what lies outside the system and how the system interacts with the outside world, e.g.:
  • Nature and characteristics of external entities (e.g. actors, roles and duties)
  • Nature and characteristics of external interfaces
  • Impacts of the system on its environment
  • Also contains the instantiated Concrete Domain Model
Guidelines & Process

Example of a Context View (source: IoT-A deliverable D1.5)

- Purchase of time-parking tickets
- Resident-parking registration
- Monitoring Centre
  - Time parking Info.
  - Registered residents
- Parking-White-List
- Control Centre
- Parking enforcement
- Registry office
- Updates from PDM
- Downloads to Control Centre
Guidelines & Process

Example of Instantiated Domain Model (source: IoT-A deliverable D1.5)
Guidelines & Process

Example of Interaction use-case (source: IoT-A deliverable D1.5)
Guidelines & Process

Requirement elicitation

• Requirement Collection is of paramount importance: it identifies what the system does and how
• The IoT ARM provides a long list of UNIfied requirements (UNIs) to start with
  – They can be adopted, modified, extended according to architect’s own objectives
  – They are available at the IoT-a website
• The UNIs provides a template inspired from the well-known VOLERE template (excel sheet)
• Requirement mapping allows to make a preliminary mapping of:
  – functional requirements to Views, Functional Group and Components
  – Non-functional requirements to Perspectives
• …resulting into a preliminary functional description and list of non-functional objectives (captured via perspectives)
• Requirement mapping (and the associated fully filled-in VOLERE template) gives an excellent basis for cross-check and ensuring none requirement is left unattended (completeness check)
• Risk Analysis for identifying threats and then issuing Security specific requirements
Guidelines & Process

Derive other Views: Functional, Information and Deployment Views

• Functional View:
  • Functional “logical” decomposition following the Functional Model
  • High level interface

• Information View:
  • Description of Virtual Entities following the Information Model
  • Service Descriptions
  • Modelling and handling of information: e.g. Ontologies and Stores
  • Flow of information between FC’s
  • Sequence Charts

• Deployment View:
  • Concrete components and their mapping to the logical FC’s
Guidelines & Process

Example of a Functional View (source: FIESTA deliverable D2.4)
Guidelines & Process

Example of an Instantiated Information Model (source: IoT-A deliverable D1.5)

- **Passive Digital Artefact (Virtual Entity): entry in the Parking White List**
  - Network Resource: Database
  - On-network VE-level IoT Service: software application for mining and updating the database

- **License plate numbers**
  - Values for the cars of the parkers

- **ValueContainer: License Plate 1**
  - Value 1
  - # MCL 1234

- **ValueContainer: License Plate n**
  - Value n
  - # MJW 5678

- **Parker type**

- **Parking zone**
  - ValueContainer: Parking zone
  - Value 1
  - # Zone A

- **ValueContainer: Parker type**
  - Value 1
  - # Nightoaker 127
Guidelines & Process

Applying Architectural Perspective to Architectural Views

- Evolution & Interoperability
- Performance & Scalability
- Trust, Security & Privacy
- Availability & Resilience
- Functional
- Information
- Deployment
- Other Views
Guidelines & Process

Applying perspectives to Views: Design Choices (DC’s)

- Perspectives propose generic “Tactics” (or strategies) that can help reaching the desired non-functional system quality
  - E.g. Replication for the Performance & Scalability Perspective
- Each tactic can be declined in different variants called Design Choices depending on which View they are applied to

<table>
<thead>
<tr>
<th>Tactic</th>
<th>Impact on Views</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replication</td>
<td>Replication of functional components (DC PS.1)</td>
</tr>
</tbody>
</table>
Structure of presentation

1. IoT Architectural Reference Model
2. IoT Reference Model
3. Guideline & Process

4. Lesson Learnt

5. Home Work
Lessons learnt

• Benefits of using the ARM
  o Excellent cognitive aid: it provides a common grounding (language and concepts)
  o Many views, viewpoints and perspectives to start with
  o Provides detailed support and guidance to different groups of people at different phases of the project in a consistent way
  o UNIs and Threat Analysis
  o Designing system use cases (Info view) happened to be a very fruitful and time saving exercise
  o Comprehensive list of components and interfaces (Annex C) to start with, however **new ones are often needed**
  o Provides complete sets of examples
  o Provide ontologies as one “implementation” of the Information Model
  o Provide comprehensive annexes with detailed interfaces, system use-cases and sequence diagrams which can/should be reused
  o However some aspects can be improved: Communication and T/S/P Models, Associations, DC’s
  o Service Centric vs. Resource Centric
Lessons learnt

• Other general benefits of using the ARM
  o Maintain consistency between various phases of product development and foster inter-team communication
  o Provides some reverse mapping e.g. ETSI M2M, EPC Global,…
  o Follows best practices in system architecture (Rozanski & Woods’ work)
  o Facilitates comparison of different architectures
  o IoT-A deliverables also provide reference implementations of some FC’s
Lessons learnt

Envisaged next steps

• How can projects contribute to the ARM
  o Project focussing on a particular system quality e.g. Interoperability or Security can come up with much detailed tactics and design choices
  o E.g. FIESTA will provide a comprehensive list of Design and Technology choices, Ontologies and Functional Components for supporting Semantic Interoperability

• ARM Profiles
  o The IoT ARM gives good guidance but leaves lot of freedom to the architect
  o Profiles aim at restricting this freedom in order to make sure a quality property is eventually reached
  o Being profile compliant means adopting very well specified interfaces, FC’s, design and technology choices

• Towards IT support
  o At the moment the main IT support is through the use of UML notation
  o Providing supporting tools to the IoT ARM is part of the Smart, Safe & Secure Platform (S3P) project roadmap

• Towards standardisation:
  o The IoT ARM is the most advanced architectural framework for the IoT. It mobilized 15+ researcher during 3 years and was used by many European projects, FP7, SmartCity, FIRE
  o The IoT ARM is being considered as major input by few standardisation bodies including IEEE P2413 working group

• Survey and summary of lessons learnt (for both COSMOS and FIESTA-IoT projects) due mid 2016
Structure of presentation

1. IoT Architectural Reference Model
2. IoT Reference Model
3. Guideline & Process
4. Lesson Learnt
5. Home Work
Homework

• Contact details:
  o Dr François Carrez, University of Surrey & Chair of Working Group „Architecture and interoperability“ @ IoT Forum
  Email: f.carrez@surrey.ac.uk; francois.carrez@gmail.com

• More Information
  o [www.iot-a.eu](http://www.iot-a.eu) for general access to „some“ of the deliverables
  o For information about Resolution infrastructure (D4.3) please ask me directly.
  o Springer Book (Open Access) “Enabling Thinks to Talk – Designing IoT solutions with the IoT Architectural Reference Model”
  o IoT-A Deliverable D1.5 (with all annexes)
  o IoT Forum web site: [www.iotforum.org](http://www.iotforum.org)
  o IoT Week web site: [www.iot-week.org](http://www.iot-week.org) (31st of May / 2nd of June in Belgrade – Serbia)
  o FIESTA-IoT website: [www.fiesta-iot.eu](http://www.fiesta-iot.eu)
  o COSMOS website: [www.iot-cosmos.eu](http://www.iot-cosmos.eu)

  o Acknowledgement: The IoT ARM is the result of intense work carried out by a large international team comprising: SAP, NEC, Alcatel Lucent, Siemens, Hitachi, Fraunhauser IML, CEA, CFR, Uni. di Roma, Uni. Würzburg and Uni. Of Surrey (Architecture Lead). The IoT ARM is a main outcome of Internet of Things – Architecture (IoT-A) FP7 EU project
Views: A view is a representation of one or more structural aspects of a reference architecture that illustrates how the reference architecture can be adopted to address one or more concerns held by its stakeholders.

Viewpoint: A collection of patterns, templates and conventions for constructing one kind of view

Perspectives: The issues addressed by perspectives are the nonfunctional requirements of the architecture

{Views, Perspectives} lead to Design Choices
Design Choices in Best Practice / Guidelines
About Views, Viewpoints and Perspectives

Perspectives

• The issues addressed by perspectives are the **nonfunctional requirements** of the architecture

• The stakeholder requirements clearly show a need of addressing non-functional requirements (~30 non-functional requirements related to systems)

• “Architectural perspective is a collection of activities, checklists, tactics and guidelines to guide the process of ensuring that a **system exhibits a particular set of closely related quality properties that require consideration across a number of the system’s architectural views.”** [Rozanski, 2005] (Definition used in D1.3)

• Tailored the approach from Rozanski et. al. to IoT Systems
<table>
<thead>
<tr>
<th>UNI ID</th>
<th>Requirement Type</th>
<th>Category</th>
<th>Description</th>
<th>Rationale</th>
<th>View</th>
<th>Perspective</th>
<th>Functionality Group</th>
<th>Functional Component</th>
<th>Domain Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNI.001</td>
<td>Non-functional Requirements</td>
<td>Privacy</td>
<td>The system shall provide a means to allow people to use Internet of Things services anonymously</td>
<td>&quot;Citizens want to protect their private data&quot;</td>
<td>Functional</td>
<td>Security and Privacy</td>
<td>Security</td>
<td>Identity Management</td>
<td>User, Service, Resource, Device</td>
</tr>
<tr>
<td>UNI.002</td>
<td>Non-functional Requirements</td>
<td>Privacy, Usage</td>
<td>Users have control how their data is exposed to other users</td>
<td>&quot;Citizens want to protect their private data&quot;</td>
<td>Functional</td>
<td>Security and Privacy</td>
<td>Security</td>
<td>Authorisation</td>
<td>Human User, Service, Resource</td>
</tr>
<tr>
<td>UNI.003</td>
<td>Non-functional Requirements</td>
<td>Self-description, Semantics</td>
<td>The system shall enable the provision and exchange of semantics between services in order to support the design of new applications</td>
<td>&quot;I would like a way to create and exchange semantics between objects in order to design new applications&quot;</td>
<td>(none)</td>
<td>Evolution and Interoperability</td>
<td>(none specific)</td>
<td>(none specific)</td>
<td>Service, Resource</td>
</tr>
<tr>
<td>UNI.004</td>
<td>Non-functional Requirements</td>
<td>Self-description, Semantics</td>
<td>The system shall enable the semantic description of physical entities</td>
<td>&quot;I would like a way to create and exchange semantics between objects in order to design new applications&quot;</td>
<td>Information</td>
<td>(none)</td>
<td>(none specific)</td>
<td>(none specific)</td>
<td>(none)</td>
</tr>
</tbody>
</table>
Functional View

Example of a Functional View (source: COSMOS deliverable D2.3)
System Use-Cases

Example from IoT-A (source: IoT-A deliverable D1.5)