Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master Thesis

# Modeling Context-Aware Systems using TOSCA

Marta Martín Sánchez.

| | |
|---|---|
| **Course of Study:** | INFOTECH |
| **Examiner:** | Prof. Dr. Frank Leymann |
| **Supervisor:** | Kálmán Képes, M.Sc. |
| **Commenced:** | January 19, 2017 |
| **Completed:** | July 19, 2017 |
| **CR-Classification:** | D.2.2,D2.11,D.2.12,D.4.7,H.5.3,I.2.8 |

# Abstract

Cloud Computing is widely accepted as a provider of virtual resources over the Internet, it is due to its economical and technical benefits, such as on-demand self-service, resource pooling and rapid elasticity capabilities. To exploit these properties reliably, it is needed to automate their internal processes for application provisioning, configuration and management. One of the standards that has been developed in the last years with this aim is the Topology and Orchestration Specification for Cloud Applications (TOSCA) by OASIS. TOSCA is a standard which enables application developers to describe applications and their management as models that incorporate components and their relations among each other. Due to the emerging of new fields such as the Internet of Things, Industry 4.0 and the upcoming Fog Computing paradigm, which depend more and more on dynamically changing situations and therefore reconfiguration of applications in such a scenario, a systematic modelling approach which handles situational dependencies directly in the models of these applications is needed. The goal of the thesis is to identify suitable modeling concepts from the field of context- aware systems and apply one of these on the TOSCA language to incorporate the inherent nature of change of future applications in the field of Cloud/Fog Computing, Internet of Things and Industry 4.0.

# Contents

1 Introduction 13
1.1 Scope of the work . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
1.2 Thesis Organization . . . . . . . . . . . . . . . . . . . . . . . . . 14

2 Fundamentals 17
2.1 Cloud Computing . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
2.2 Business Process Modeling . . . . . . . . . . . . . . . . . . . . . . 18
2.3 Context Aware Systems . . . . . . . . . . . . . . . . . . . . . . . 21
2.4 TOSCA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

3 Related Works 41
3.1 Integration of Situation-Awareness in Workflow Models: SitOPT . . . . . 41
3.2 Contextualization of Business Process . . . . . . . . . . . . . . . . . 42
3.3 Integration of Imperative and Declarative Approach to model Cloud Applications . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44

4 Motivating Scenario 47
4.1 Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
4.2 Modeling requirements . . . . . . . . . . . . . . . . . . . . . . . . 53

5 Concept 55
5.1 Concepts Context Modeling . . . . . . . . . . . . . . . . . . . . . . 55
5.2 Concepts Imperative Approach . . . . . . . . . . . . . . . . . . . . 56
5.3 Concepts Declarative Approach . . . . . . . . . . . . . . . . . . . . 67
5.4 Mapping . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76

6 Conclusions and future works 87

Bibliography 89

# List of Figures

2.1 Representation of basic components BPMN diagrams . . . . . . . . . . . 19
2.2 Overall Architecture of CAPEUS . . . . . . . . . . . . . . . . . . . . . . 25
2.3 Structure Mark-up Model . . . . . . . . . . . . . . . . . . . . . . . . . . 26
2.4 User Agent Profile . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
2.5 Example of graphical models . . . . . . . . . . . . . . . . . . . . . . . . 29
2.6 Different types of facts [SL04] . . . . . . . . . . . . . . . . . . . . . . . 30
2.7 Overview service template topology by TOSCA . . . . . . . . . . . . . . 35
2.8 Conceptual Layers of TOSCA-based applications . . . . . . . . . . . . . . 36
2.9 Example of a simple Build Plan . . . . . . . . . . . . . . . . . . . . . . . 37
2.10 Imperative and Declarative Approaches . . . . . . . . . . . . . . . . . . 39

3.1 Three steps of Generating Provisioning Plans [BBK+14] . . . . . . . . . 45

4.1 General Description of Motivating Scenario . . . . . . . . . . . . . . . . 48
4.2 Representation of the first Use Case . . . . . . . . . . . . . . . . . . . . 49
4.3 Representation of the Use Case . . . . . . . . . . . . . . . . . . . . . . . 50
4.4 Representation of the third Use Case . . . . . . . . . . . . . . . . . . . . 50
4.5 Detection bad performance . . . . . . . . . . . . . . . . . . . . . . . . . 51
4.6 Representation of Forth Use Case . . . . . . . . . . . . . . . . . . . . . . 52
4.7 Representation of Fifth Use Case . . . . . . . . . . . . . . . . . . . . . . 52

5.1 Monitoring Workflow . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
5.2 Imperative General Model Approach of 1st Use Case . . . . . . . . . . . 58
5.3 Expanded View Fetch VMs activity . . . . . . . . . . . . . . . . . . . . . 59
5.4 Expanded View Provide Service Activity - Fixed Node Template . . . . . 60
5.5 Expanded View Provide Service Activity - Not Fixed Node Template . . . 60
5.6 Imperative General Model Approach of 2st Use Case . . . . . . . . . . . 61
5.7 Expanded View Connection/Disconnection appropriate Gateway . . . . . 62
5.8 Imperative General Model Approach of 3rd Use Case . . . . . . . . . . . 63
5.9 Imperative Alternative Model Approach of 3rd Use Case . . . . . . . . . 64
5.10 Imperative Alternative Model Approach of 4th Use Case . . . . . . . . . 65
5.11 Imperative Alternative Model Approach of 5th Use Case . . . . . . . . . 66
5.12 Declarative General Model Approach of 1st Use Case . . . . . . . . . . . 68
5.13 Possible states of 1st case topology during run-time . . . . . . . . . . . . 68

5.14 Declarative Alternative Model Approach of 1th Use Case . . . . . . . . . 69
5.15 Possible states of 1st case(alternative) topology during run-time . . . . . 70
5.16 Declarative General Model Approach of 2nd Use Case . . . . . . . . . . . 70
5.17 Possible states of 2nd case topology during run-time . . . . . . . . . . . 71
5.18 Declarative General Model Approach of 3rd Use Case . . . . . . . . . . . 71
5.19 Possible states of 3rd case topology during run-time . . . . . . . . . . . . 72
5.20 Declarative Alternative Model Approach of 3rd Use Case . . . . . . . . . 72
5.21 Possible states of 3rd case(alternative) topology during run-time . . . . . 73
5.22 Declarative General Model Approach of 4th Use Case . . . . . . . . . . . 74
5.23 Declarative General Model Approach of 5th Use Case . . . . . . . . . . . 74
5.24 Possible states of 4th and 5th topology during run-time . . . . . . . . . . 74
5.25 General Architecture . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
5.26 1 Concept: Data connector between Processing Logic and unique Data Resource . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
5.27 TOSCA realization with explicitly modeled data query . . . . . . . . . . 81
5.28 Data Connector with Operations Applied to Multiple Data Resources . . 82
5.29 TOSCA realization with join operation assignment via join interface . . . 84
5.30 Integrated architecture for Location Aware System . . . . . . . . . . . 85

# List of Tables

2.1 Evaluation of Context Models . . . . . . . . . . . . . . . . . . . . . . . . . 33
2.2 Evaluation of approaches [BBK+15a] . . . . . . . . . . . . . . . . . . . . . 40

# List of Listings

2.1 Example of Key Value Model . . . . . . . . . . . . . . . . . . . . . . . . . 24
2.2 Example of Mark-up Model . . . . . . . . . . . . . . . . . . . . . . . . . 26

5.1 Processing Logic Node Type Definition . . . . . . . . . . . . . . . . . . . 79
5.2 Context Resource Node Type Definition . . . . . . . . . . . . . . . . . . . 79
5.3 Data Connector Type Definition . . . . . . . . . . . . . . . . . . . . . . . 80
5.4 Context Collection Node Type Definition . . . . . . . . . . . . . . . . . . 83

# 1 Introduction

The spread of recent technological advances is changing the way in which we consider Internet and the Computing Paradigm [YMCL13]. Cloud Computing has introduced a wide range of possibilities and capabilities in developing applications. Moreover, Pervasive Computing has introduced a new perspective which involves the introduction of technology in the people‘s daily life by the penetration of mobile computing shown as “invisible” [Sat01] which allows to recognize automatically changing situations providing the basis for automated adaptation of process executions. This involves a high impact on modeling and development of applications in which context information is more and more valuable. For that reason, a new type of systems are taking more relevance in this area, which are so-called Context Awareness Systems, by taking advance of the context information to adapt its performance and react to changes of its close environment. This means that the applications would be adaptive to different situations and as one of the objectives of Industry 4.0, the time of reaction would be insignificant. These systems are characterized by having a dynamic architecture which simplifies end-to-end processes and reduces costs and by making use of Big Data to generate knowledge in order to be aware of the context. A wide range of areas can take advantage of these systems, from emergency or healthcare areas which need to monitor the context to react as quickly as possible, to IT areas in which improving the quality, decreasing the impact of points of failure and adapting its performance to the users situation are a valuable advantage from competitors.

Modeling is considered as a feedback process, models are in constant improvement, continuous examination and adaptation. It is not only required to this type of applications to be aware of the data associated with the environment, but also a high level of situation understanding and adaptation of the states of the system. One of the most benefits of this approach is the adaptability which have the topologies to be reused in different applications context. Moreover, it can be used in many different levels of granularity and this type of topologies offer an assistance to stakeholders to handle organizational change.context, which is the aim of this thesis. [End95; KBS+16]

Modeling context is a challenging task. First authors who wrote about context-awareness only take into consideration location changes when describing context. Although many definitions have been given of this term, an accepted one include all the factors that can characterize a situation such as temporal, spatial, environmental, device, network information or communications capacities. Many modeling approaches have been used: simple structures which define the context attributes which are taken into account such as key-value or mark-up models; or more complex structures which introduces rules, relationships, logic between the different context entities such as ontology models.

Cloud Applications are required to be automatically deployed and managed, portable between different environments and its components should be interoperable and reusable. This requirements can be fulfilled by the standard Topology and Orchestration Specification for Cloud Applications (TOSCA). This standard defines application topologies and management plans in order to offer a full-automated management of applications. TOSCA supports two different processing approaches: imperative approach which makes use of Provisioning Plans that precisely describe provisioning tasks to be executed enabling the customization of the service, while declarative approach adapts application topology models by a Runtime Environment aware of provisioning logic reducing the effort to model the application behaviour.

Due to the benefits of TOSCA modeling, TOSCA could be considered as a candidate for modeling context aware systems. However, data management and reaction capacities to changes are not specified in TOSCA. For this reason, it is necessary to develop a general architecture in TOSCA to model these type of systems.

## 1.1 Scope of the work

The objective of this master thesis is to model a solution to integrate context-aware systems into TOSCA. It would be necessary to understand and fulfill the requirements that this type of applications have and evaluate about how they can be modeled. After this analysis, we will define a general solution by modeling how the application structure and integration into TOSCA will be handled by constructs of context awareness. To analyze the possible profits and deficiencies a possible scenario will be described and modelled with this solution. The conclusions of this work would help to guide forthcoming implementation.

## 1.2 Thesis Organization

This document is structured as follows:

**Chapter 2 – Fundamentals:** In this chapter we described the fundamentals concepts for this thesis such as Cloud Computing and Context Aware Systems.

**Chapter 3 – Related Works:** In this chapter we introduce similar works of modeling context aware systems.

**Chapter 4 – Motivating Scenario:** In this chapter we describe a scenario which will be used to present and evaluate the modeling solution proposed, analyze different possibles use case developed in this scenario and finally specify the requirements needed to model the scenario.

**Chapter 5 – Concept:** In this chapter we introduce our conceptual modeling approach to the scenario evaluating the suitability of using imperative or declarative approaches in the use cases proposed and how our concept could be mapped into TOSCA.

**Chapter 6 – Conclusions and future works:** - In this chapter, we summarize our work and analyze if the objectives have been fulfilled. while describing possible future improvements.

# 2 Fundamentals

This chapter introduces the fundamentals concepts related to this thesis which ranges from Cloud Computing to Context Aware Systems and TOSCA .

## 2.1 Cloud Computing

Cloud Computing has been defined by the National Institute of Standards and Technology (NIST) as "*model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications or services which can be rapidly provisioned and released with minimal management effort or service provider interaction*"[MG11].

Since Cloud Computing has been introduced, it has changed the way we use the computing infrastructures by enabling an access to a wide range of configurable computing resources. This technology offers virtually unbounded storage and processing competences allowing a rapid scalability with minimal management and cost effort of applications because of the on-demand nature of Cloud Computing. In order to gain both economical and technical benefits, large companies such as Amazon, Google, Facebook have embraced this paradigm for distributing services over the Internet [Dis15].

Because of the advantages of Cloud Computing, recent applications which involve mobile devices make use of the computing, memory and storage resources as well as the rapid scalability that is offered by cloud servers. Moreover, context is getting more and more valuable to smart applications in order to adapt its behavior to the current necessities of the users. This implies complex processing which can be reduced by using Cloud Computing. The proliferation of this kind of smart applications have arisen a new area of development, known as Mobile Cloud Computing [NPB13]. Many challenges have been introduced such as integration and interoperability requirements, and the need of understanding, anticipating and adapting the behavior of applications without the intervention of users. To fulfill the requirements of this paradigm, it involves taking care of a good architecture, an effective management of context information, the communication and coordination between the entities which lead to the smart adaptive behavior requirement [BDPP16].

As a consequence of the amazing increment of different Internet-enabled devices and the ubiquitous wireless and cellular data networks, a wide range of users are requiring customized services. It is believed that the next step in Cloud Computing paradigm would be outside the dimension of traditional cloud services, such as context aware services [DPK+09; WB10; YD16].

## 2.2 Business Process Modeling

The objective of Business Process Modeling is to provide high-level specifications taking into account the business requirements which can be automatically simulated, validated and turned into a real-time application. With the use of a process modeling standard, the understanding is not only tied to users of a particular tool, but also to all the users of the standards because of its multi-vendor definition. To model business processes, it is not only necessary to include the sequence of process activities, but also information required to transform and optimize company's business processes [Fis06; Sil11].

Despite different approaches in modeling used such as textual or tabular descriptions [Dam07], the widely accepted models are the graphic model diagrams. Although many standards are used in process modeling such as XPDL (XML Process Modeling Language) which is used to define automatic processes and BPEL(Business Process Execution Language) used to execute processes, BPMN has become the leading standard for business process modeling. BPMN is an expressive language able to describe the behavior of process models in an expanded way, and later specify the technical details which control the process execution. BPMN could be considered as a bridge between the business and IT world, introducing an standard that could be used by both sides. The original aim of BPMN when it was developed by the Business Process Management Initiative (BPMI) was to provide a graphical notation for process description by the use of BPML(Business Process Modeling Language) which was replaced by BPEL. Few versions have been introduced in this standard, but the most important and recent one is the so-called BPMN 2.0 which have introduced different new diagram types(collaboration, choreography, conversation) to cover different types of applications. In addition, a standardized exchange format for BPMN models in order to do them portable in different tools. Moreover, execution semantics and capabilities to introduce executable process designs were created to automate the deployment of the processes [All16].

BPMN describes the processes as a sequence of activities which leads from an initial state to a defined final state, including all the possible paths from this two states. An activity in BPMN is considered as a unit of work performed repeatedly in the course of the business, but each instance refers on a different piece of work with a well-defined start and end state.

In the Figure 2.1 the most important described components of a BPMN diagram are represented.

The activities can be grouped in sub-processes which can be represented in the diagram in a compressed or an expanded way depending of the level of detail that it is desired. In the left side of the figure, both views are represented. These sub-processes are characterized for requiring an start event and an end-event and allowing to have intermediate events which can interrupt the execution of the activities of the sub-process.

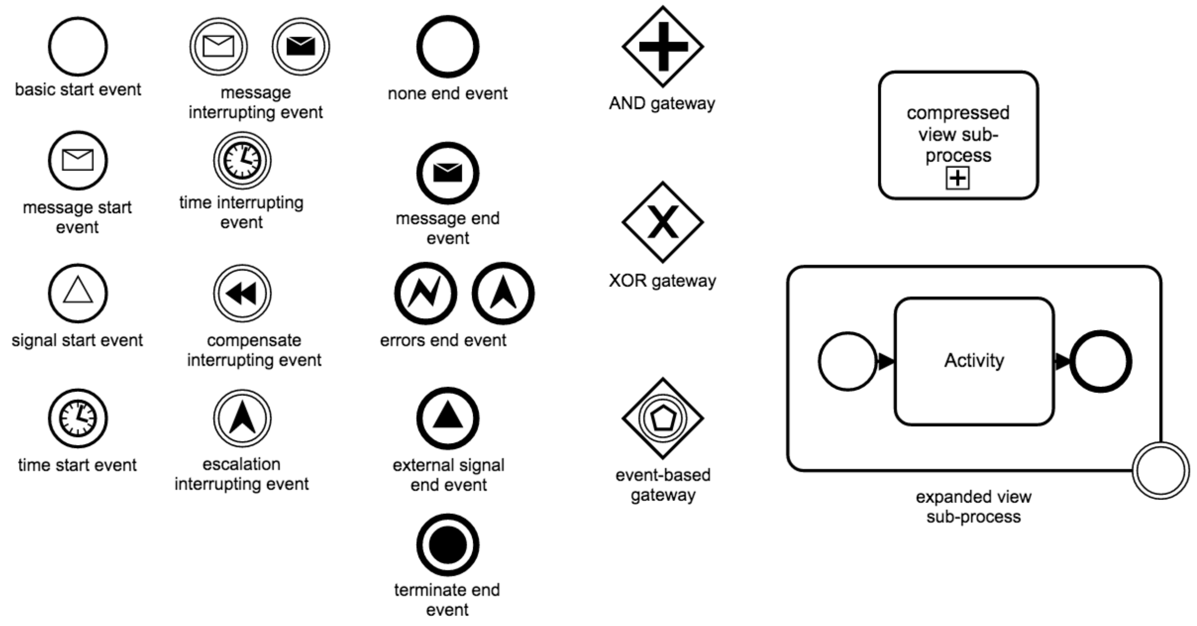


**Figure 2.1:** Representation of basic components BPMN diagrams

To alter the control flow of a process, BPMN introduces decision points. The most important decision gateways, which are represented in BPMN as rhombuses are the following:

- **AND gateways**: These gateway trigger the parallel execution of segments of the workflow. By that, it does not mean that they are executed at the same time, it means that it does not matter which is executed first.
- **XOR gateway**: This gateway, as it will be seen in the chapter 5, will be essential for declarative modeling to select the appropriate node instances to be installed in the topology to the context situation in which the system is. As a difference with the previous gateway, it only selects one path to be executed from the possible paths.

- **Event-based gateway**: This gateway is aware of the events that are triggered in the process instance and depending on which is executed first the path related would be executed. This gateway will be used very often in the imperative modeling when the changes of the topology of the system are related to changes in the context (time-frames, performance of servers,..)

Three different types of events are used in the workflows defined by this standard:

- **Start Event**: All process or sub-processes are initiated with this type of event, which are introduced as a single ring in the diagrams. It represents what triggers the execution of an instance of the process. By contrast with other types of events, in each process there is only one start event. There are a wide range of types of start events: from basic start events to trigger events such as receiving a message, time scheduling or external signals, all of them represented in the first column of the right side of the Figure 2.1
- **Intermediate Events**: This type of events, represented with a double ring in the diagrams, occur during the execution of the process instance. These events can be of two types: interrupting events which stops the execution of the activity if it is not completed and execute the exception flow or non-interrupting events represented with a dashed double ring which triggers a parallel path of the process when the activity is completed. As well as the start events, message reception, time scheduled and external signal triggers are different types of intermediate events, but in addition new kinds are introduced such as error or compensation triggers. We are going to use mainly interrupting events to model the scenario proposed in the Section 5 so in the Figure 2.1 these events are represented in the second column of the right side. There is also possible that intermediate events can be generated by the process, so-called catch events represented by filled-icons.
- **End events**: These events, represented as a ring with a single thick border, are which indicates the end of the execution of the instance of a process or a sub-process model. There are as many end events as paths that a process or sub-process has. As a difference with the start event, these events are generated by the process, represented in the diagram by a filled icon inside the ring. The most representative end events which are represented in the Figure 2.1 are the none event, message sending event and terminate event.

Although BPMN is not originally adapted to context-awareness, the introduction of improvements in the version 2.0, such as data flows, and the elements, previously described; it is a perfect candidate to model these types of behavior. Recent researches have introduced solutions of modeling context awareness systems by using this language. Julian Dörndorfer et.al [DS17] proposed an extension of the BPMN enabling the modeling of mobile context sensitive business processes by introducing a new type of events related to context variables. Alaaeddine Yousf et. al have also proposed an extension so-called UBPMN [YBSD16] with the aim of creating end-to-end ubiquitous business processes and guarantees their portability in a conservative way by introducing tasks and events structures associated to the context.

## 2.3 Context Aware Systems

Many definitions have been used to describe context recently, some of them so general that they include all conditions that exist in a given situation, and definitions that are more specific capturing concrete factors such as location (where), type of device (what) or identity (who). Both types have its limitations, the first type is really vague and does not specify the focus and the difficulty of implementation; and the second one may not capture all valuable events or facts which has influence in the situation. In 2000, a new definition of context was introduced which by trying to include the different perspectives in which the context was defined before [AG99].

*"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."*

With the use of this definition, entities, which are every necessary element included in the context (including the user and the application itself), have vital importance between the application and the user. The perspective in which context is seen, instead of considering it as a central structure, as interactions between entities. This new definition allows designers to capture all important elements of the context regardless of which perspective is taken by the design team. It provides a way with this perspective of evaluating and validating the possible elements which take part in the context. Nevertheless, it is still very difficult to include all aspects which may influence the context. Moreover, the dynamic property which characterizes context provokes that the importance of it would depend on the situation [AG99; BE04].

Context aware systems are taking more and more presence in Pervasive Computing. The most important characteristic of this type of systems is the ability to sense the context in the environment and adapt its operations to predefined or dynamic changes without the intervention of users. By this, more efficiency, usability and effectiveness are achieved.

With the big impact of mobile devices, it is highly desirable that services and applications could adapt its behavior to their current location, time, type of device, and other environment factors. Moreover, context data might change quickly so it requires an intensive monitoring. This data can be taken by different sources such as sensors, network status and device information [PFCP10].

These types of systems are not relatively new. Since 1992, some examples of context aware systems have been developed. The first ones were focused on determining the location of the users because of being the most frequent context attribute [BCB+10]. However, in the last few years other attributes have been taken into account.

There are a wide range of areas of application of these kinds of systems such as the healthcare area in which monitoring is important to improve the performance of the staff of the hospital; or emergency situations such as earthquakes or floods in which the prevention and a efficient first aids are essential. Moreover, areas like the transport or industry would have a high impact with this new type of systems improving its performance and efficiency, introducing an additional value and adapting to the necessities of every one. Not only are these kinds of systems improving existing areas, but also creating new applications such as intelligent homes in which context information helps to improve the conformability of the people who live there, improve efficiency and reduce costs.

One of the challenges these types of systems have to continue working with, is security. It is necessary to guarantee the privacy, the integrity and the confidentiality of the context data. A fault in security can provoke that the system use context data that has been manipulated or sensible data such as data of intelligent homes can be analyzed by intermediates causing harmful risks.

Context-aware systems can be implemented in many ways depending on specific requirements, conditions and the way these systems acquire the contextual information [DR07]. A good context data modeling is really valuable because it can reduce the complexity of these applications, improve their maintainability and evolution. In the section 2.3.1 we describe different modeling approaches that have been researched during the last years in order to manage and integrate context in a wide range of applications.

According to Knappmeyer et. al [KKR+13] context can be classified into 9 categories:

- **Spatial Context**: This type of context is related to information about the physical position of an element and the relative position to other elements (proximity).
- **Temporal Context** : This type of context is related to information about the absolute or relative point in the time of an event.
- **Device Context**: This type of context is related to information about the capabilities and resources of devices which interacts with the system.
- **Network and Communication Context**: This type of context is related to information about network characteristics such as WiFi access point or bandwidth available.

- **Environmental Context**: This type of context is related to information about the physical surrounding of an entity such as noise level, humidity, pollution or light intensity.
- **Individuality and user profile context**: This type of context is related to information about identified users such as preferences, interests and habits.
- **Activity Context**: This type of context is related to information about which task an entity is executing and which are planned.
- **Mental Context**: This type of context is related to information about the physiological status of the user such as level of stress, feelings and so on.
- **Interaction Context**: This type of context is related to information about the interactions in the system, not only between the application and the user, but also interaction between users.

### 2.3.1 Context Model

A context-aware system, in order to access to the context, needs a well designed model as it was said in the section 2.3. Context sharing and interoperability of applications are the objective of current research so uniform context models, representation and query languages as well as reasoning algorithms are being developed to facilitate this aim. To develop any kind of modeling approach it is necessary to take into account the following requirements of ubiquitous computing:

1. **Requirement 1: Distributed Composition**: A context model should handle high dynamics in terms of time, network topology and sources in the composition and administration of context models and its data.
2. **Requirement 2: Partial Validation** :A context model should give partially validation of contextual knowledge despite it is not available due to the complexity of contextual interrelationships, which makes any modeling error-prone.
3. **Requirement 3: Richness and Quality of Information**: A context model should support quality and richness indication due to variations of the quality of the measures taken and in the abundance supported by a wide range of sensors.
4. **Requirement 4 : Incompleteness and ambiguity**: A context model should handle incompleteness and ambiguity in the amount of information available at any time using techniques such as interpolation of incomplete data on the instance level.

5. **Requirement 5: Level of Formality**: A context model should describe facts and interrelationships in a precise and traceable manner and all entities which composes a system should share the same interpretations of data exchanged.
6. **Requirement 6: Applicability to existing Environments**:A context model should be applicable within existing the infrastructure of ubiquitous computing environments.

For that reason, in the past different kinds of context models have been subject of research. Initially, the context models were related to only one application or an application field, but lately generic models are more taken into account because of the benefits to a high range of applications. In this section, we present the most relevant context models and evaluate the usefulness according to the different kinds of context presented in the previous section.

### Key-value models

The model of key-value is considered the simplest data structure for context modeling. To describe the context, it is given a list of attributes composed each of them by a context key and a context value. These pairs are usually realized as environment variables. In the Listing 2.1 it is presented a simple example of the Key-Value Model where the corresponding instances describe contextual information about a location.

**Listing 2.1** Example of Key Value Model

```
Model:

[<ContextKey> Context Value]*

Instances

<Location_lat>48.745853; <Location_lon>9.105398;

<Temperature>1 <Temperature_unit>Centigrade;

<Humidity> 36.7; <Humidity_unit>\%;
```

It is commonly used in distributed service frameworks in which the key-values pairs are used to describe the capabilities of a service. Then the discovery procedure operates an exact matching algorithm using this key-value pairs. In that way, an architecture for context-aware service provisioning has been proposed with the name of CAPEUS. Users receive, with this system, the service which fits more with them (location, aim of the user...). In the Figure 2.2, it is depicted the overall architecture of CAPEUS [SML01]

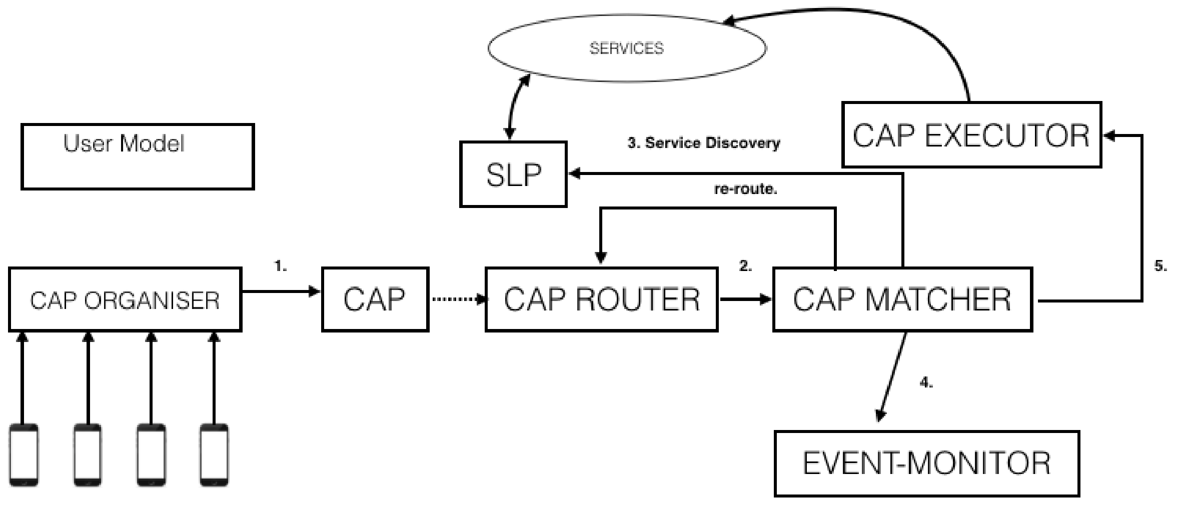


**Figure 2.2:** Overall Architecture of CAPEUS

In general, key-values are manageable, easy to implement but they lack of capabilities for sophisticated structuring for enabling efficient context retrieval algorithms [BCB+10; GFHK14; NH13].

### Markup scheme models

To represent contextual information, markup models are based on a hierarchical data structure consisting of markup tags. The content is usually defined by markup tags in a recursive way. A variety of markup languages are used for context modeling including XML. Continuing with the proposed example of the key-value models, in the Figure 2.3 it is presented the markup scheme and the hierarchical structure describing the contextual information of it [GFHK14; NH13].

**Listing 2.2** Example of Mark-up Model

```
Model:

<Location>
   <Temperature>
      <Temperature_unit> Centigrade </Temperature_unit>
      <Temperature_value> 1 </Temperature_value>
   </Temperature>
   <Humidity>
      ....
   </Humidity>
</Location>
```

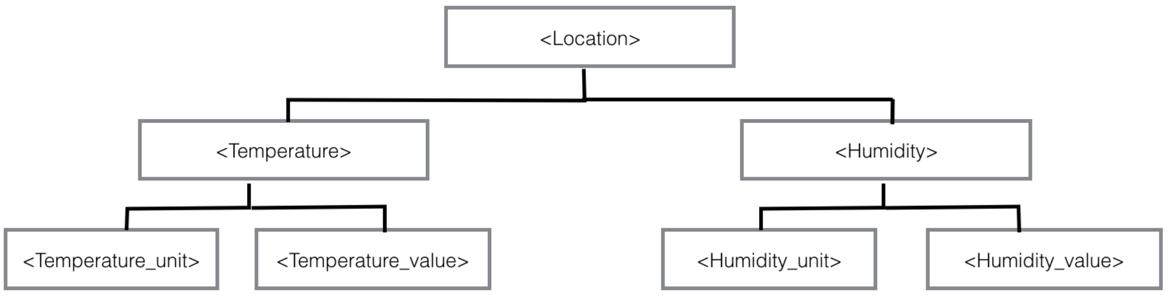


**Figure 2.3:** Structure Mark-up Model

This modeling approach is manageable and the existence of a scheme definition provides a stable foundation for developing validation tools. However, strict hierarchical structures imply lack of flexibility.

Some modeling approaches which use this technique are:

1. **CC/PP (Composite Capabilities/Preference profile)**: It is an W3C's specification which defines a basic structure for profiles, composed by a strict two level- hierarchy. This specification was the first of using Resource Description Framework (RDF) and including elementary constrains and relationships between context types. Not only is the goal of this approach to express both device capabilities, but also user preferences. However, the definition of the particular components and attributes is not done by this specification, but by other standardization bodies using RDF scheme. This flexible mechanism ensures CC/PP extension. Moreover, the XML interchange format and a referencing mechanism for external resources excellently meet the interchangeability and composability/ decomposability as requirement. However, the CC/PP framework has some disadvantages such the limited capabilities in capturing complex profiles of context structures such as

relationships and dependencies between entities because of the two level-hierarchy, limited capabilities in allowing consistency checking and supporting reasoning on context, the limitation of capturing complex profile structures, the requirement of attributes names to be unambiguous even if there are used in different components [BCB+10; CFJ03a].

2. **UAProf (User Agent Profile)**: The goal of this specification, defined by the Open Mobile Alliance (OMA) is capturing capability and preference information for WAP-enabled mobile devices. As CC/PP is, UAProf is also based on RDF. It is used generally to provide the more appropriate content for different devices based on the information taken. The information can be clustered in six categories: hardware, software, browser, network, wap and push characteristics. [TJH10]. In the Figure 2.4 it is shown what it is the schema of the UAPROF. First of all, a parent element has a unique child which it can be seen in the figure as the description which identifies the element. This element has as child all the attributes of each component: HardwarePlatform, SoftwarePlatform... but it is required that there is only one component for each profile. Each attribute is described with the rdf:Description tag.

```
<rdf: RDF.......>
  <rdf:Description......>
    <prf:component>
        <prf:Description rdf:ID="HardwarePlatform">
      ....
        </prf:Description>
    </prf:component>
    <prf:component>
        <prf:Description rdf:ID="SoftwarePlatform">
        ....
        </prf:Description>
    </prf:component>
    .....
  </rdf:Description>
</rdf:RDF>
```

**Figure 2.4:** User Agent Profile

3. **CSCP (Comprehensive Structured Context Profiles)**: Instead of CC/PP, CSCP does not impose a fixed hierarchy. In contrast, it gives flexibility by expressing natural structures of profile information (session profiles) as required for contextual

information using RDF/S. CSCP supports decomposability by means of external references which are used to extract sub-profiles to separate CSCP documents and provides features to attach conditions and priorities to attributes, extending the capabilities to express user preferences [BCQ+07; BHH04; CFJ03a] .

### Graphical models

Expressing relationships between context entities is the goal of graphical models. These models are the best for structuring contextual data. Graphical visualization is the basis of the representations, supported by many powerful graphical design tools. Moreover, it is allowed to automatically derive code representations and model entity relationships of databases. Graphical context models have commonly been used as generic structure that can be used to model context in environments. These models are in general more expressive than key models and hierarchies but they have lack of reasoning capabilities, they are not good at handling ambiguous data sets and struggle with the incomplete data and partial validation [BCB+10; GFHK14; NH13].

There are two main directions used for graphical modeling:

1. Unified Modeling Language (UML): Due to its generic structure and that it is composed by graphical elements which are excellent foundations for context modeling, UML is an appropriate approach for modeling. There are a variety of examples of use of this tool such as air traffic management, contextual information classification... [Bau03]. In the Figure 2.5 it is presented the graphical context model of the introduced simple example in the previous types of models using UML class diagrams.

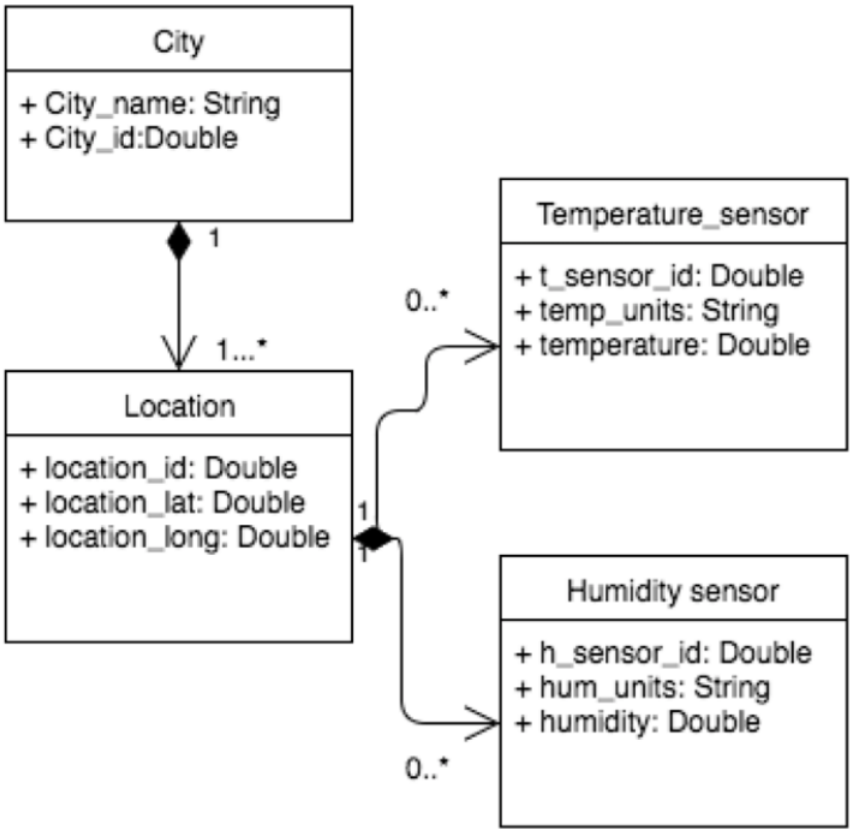


**Figure 2.5:** Example of graphical models

It can be seen in the diagram that a city which is the focus of environmental analysis is composed by different locations in which the sensors are located. Each location has an identification, as well as the city, and its GPS coordinates. In each location, different types of sensors are introduced and each one offers measures such as temperature or humidity values.

2. Object Role Modeling (ORM): The role of this technique is the identification of appropriate fact types and the roles that entity types take part in. Extensions of ORM allows to categorize fact types in static or dynamic according to its persistence, profiled, sensed or derived depending on its source. The last extension of ORM allows to introduce a new type of relationship between facts where a change in one fact triggers the change in the other fact [SL04].

   In the Figure 2.6, different types of facts are represented: the first case the two entities have a fixed relationship because a node is always attached to a type, so that means that is a static fact type. The second case, the two entities are related because when a user is registered in a system he has permissions to use specific devices so there is a profiled fact type. The third case a person is detected in a specific location e.g city so there is a sensed fact type. The forth case it is detected in a unique location a person and a device so a new relationship is created because the user is detected close to a device. This type of fact is called derived . The fifth case describes a relationship that is temporal, this means, in an specific point of time an user is doing an activity but when he finishes it, the relationship would be disabled. This type of fact is called temporal. Finally, the last case represents a dependency that if a person is in an activity also need to be located in a specific location: This type of fact is called fact dependency.

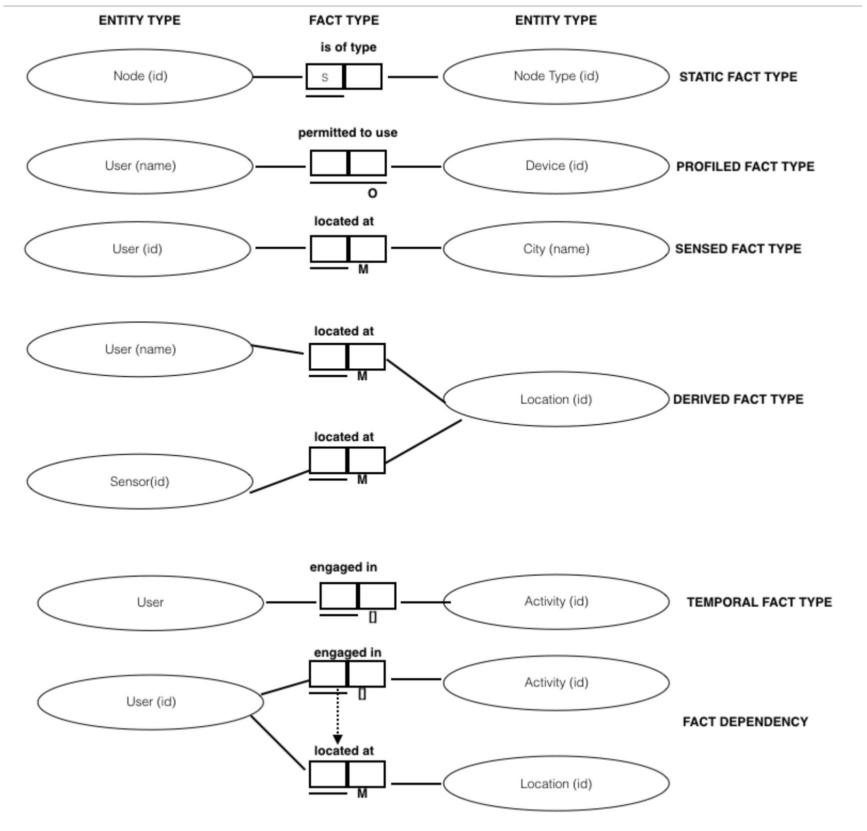


**Figure 2.6:** Different types of facts [SL04]

Object-oriented models

By using Object-oriented models, the context is classified into classes of objects and their relationships. The main benefits of the object-oriented paradigm (abstraction, encapsulation, polymorphism, reusability and inheritance) are powerfully used in this model to hide parts of the problems arising from the dynamics of the context in ubiquitous environments by encapsulating in object levels and hence hide to other components. For that reason, the access to contextual data is only possible through specific implemented interfaces. Following with the introduced example of previous model, using this technique the structure of the model would be composed mainly by two different components: humidity and temperature, which would be implemented independently and hide their implementation to the other. There are two projects in which object-oriented models have been used to model the context:

- TEAPROJECT—>“cues”:A cue is a function taking a value of a single sensor in a certain timing and giving as a result a symbolic or a sub-symbolic value. For each cue, it has a finite or an infinite set of possible values. It is possible to have different cues based on the same sensor. Modeling context, therefore, is the abstraction level on top of the available cues, which provide contextual information through their interfaces hiding the details of how it is determined the output values.
- Guide project—> Active Object Model: Its goal is to be able of managing a great variety of personal and environmental contextual information while maintaining scalability. All the details of the data collection and fusing are encapsulated within the active objects and therefore hidden to other components.

All the examples of use of object-oriented technique share the benefit of its inheritance and reutilization of capabilities, the simplification of knowledge representation in very complex domains and systems and its strength in the requirements. On the other hand, invisibility causes undermining of the formality of requirements [BCB+10; GFHK14; NH13; SML01].

### Logic based models

Context is defined, by this technique, in term of facts, expressions and rules with the highest possible level of formalism. Contextual data can be derived by existing facts, logic conditions, knowledge bases and rules, this method is known as process of reasoning or inference. This new contextual information, given as a result of this process, can be presented as a new fact in a formal way. Following with the environment example, values taken from temperature and humid can determine if it is raining or snowing for example. Different approaches for this technique have been given:

- The first logic based context modeling approach was researched and published as Formalizing Context in 1993 by McCarthy introducing context as abstract mathematical entities with properties useful in artificial intelligence **??**. Instead of giving a definition what context is, a set of simple axioms for common sense phenomena was provided, using lifting rules relating the truth in one context (axiom) to the truth in another context. Accordingly, the concept of inheritance is supported.

- A second context modeling approach, is less on context modeling than on context reasoning. Encoding a context in an individual's subjective perspective. This approach was firstly developed by McCarthy (Formalizing Context), and then by Akman and Surav (Extended Situation Theory. The wide range of different contexts is addressed in form of rules and presuppositions related to a particular point of view, representing the facts related to a particular context with parameter-free expressions supported by the situation type which corresponds to the context.

Although the formality and support for reasoning makes this context model high valuable, there are some drawbacks such as difficulty to construct implement and maintain complex logic systems which covers a wide range of contexts, lack of possibility of partial reasoning and validation and detection and solving incompleteness ambiguity and low quality of information [GFHK14; NH13; SML01].

**Ontology Models**

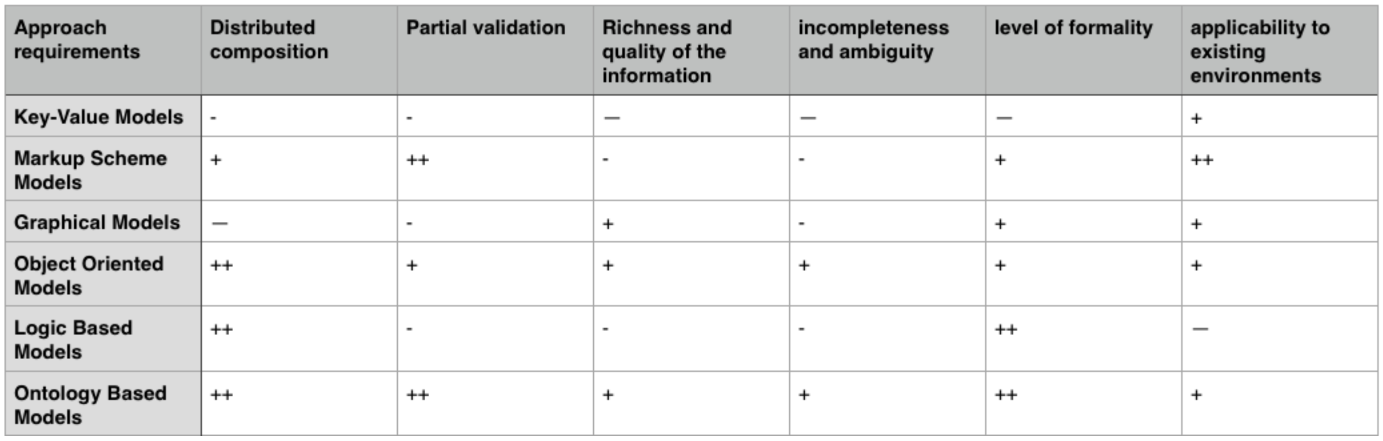
This technique is a clearly combination between logic and object-oriented modeling techniques being a promising tool to define concepts and interrelations. This approach has features to provide formalizations of real-life entities onto a data structure utilizable by computers. Context is modeled by a common vocabulary which represents knowledge about the domain: classes (concepts), individuals(facts) and properties (roles, relationships). Ontology reasoning is used to analyze and evaluate the contextual knowledge, allowing computers to determine the contextual compatibility, compare contextual facts and deduct new and more complex context from data measured. Ontology models are characterized by semantic interoperability because of a common understanding of the infrastructure, reuse of domain knowledge and ability of formal analysis such as context reasoning. However, the most context ontologies lack of generality and have not addressed important issues such as context classification, context dependency and quality of context which are main points in context reasoning. Following with the environment example, with this technique it is possible to expand the functionality of the application including new context types such as other environments measures and with the use of context reasoning deduce new information and actuate in the different locations [CFJ03b; GFHK14; GWPZ04; NH13; SML01; WGZP04]. For example, we preseent two different approaches which use ontology models:

- CONON (Context Ontology): This approach uses a two level hierarchy ontology. The upper ontology captures general concepts which are basic to all context-aware applications, and a second level is extensible to add domain-specific ontologies. To capture context information, CONON defines general concept which are believed to be the fundamental context: Computation Entity, Location, Person and Activity which not only form the structure of context, but also act as evidences into associated information. In the lower ontologies would be included what is needed for the specific context-aware application.

- CoBrA (Context Broker Architecture): In this approach, a broker-agent architecture to provide runtime support for context-aware systems supporting agents, services and devices to interact is presented. It is based on OWL (Ontology Web Language). Its principal component, context broker, provides a shared model to represent context information enabling context reasoning to provide new context information. Context data is acquired by different sources: sensors, agents and also the Web. [BCQ+07]

### Evaluation

Having in mind the requirements presented in the enumeration 2.3.1, it is necessary to analyze if the context models presented previously fulfill them. In the Table 2.1an evaluation of these models of covering these requirements is pictured.



| Approach requirements | Distributed composition | Partial validation | Richness and quality of the information | incompleteness and ambiguity | level of formality | applicability to existing environments |
|---|---|---|---|---|---|---|
| Key-Value Models | - | - | — | — | — | + |
| Markup Scheme Models | + | ++ | - | - | + | ++ |
| Graphical Models | — | - | + | - | + | + |
| Object Oriented Models | ++ | + | + | + | + | + |
| Logic Based Models | ++ | - | - | - | ++ | — |
| Ontology Based Models | ++ | ++ | + | + | ++ | + |

**Table 2.1:** Evaluation of Context Models

The strength of key-value models is the applicability in existing ubiquitous computing environments. However, it can be seen that the key-value models are the weakest on requirements mainly because of its simplicity which is beneficial for management and handling error risk but a drawback for quality meta- information and ambiguity. It is only possible on the instance level distributed composition and handling incompleteness, there is no structure available for checking and it is required difficult processes to obtain partial validation. The strength of markup models are the partial validation ability because of its scheme definition and its set of validation tools and as key-value models applicability to existing environments. Depending on each single approach, the distributed composition requirement and the formality can be met also. Handling incompleteness and ambiguity is not covered by the context model so must be implemented by each application. The strength of graphical models is the ability to describe efficiently the structure of contextual knowledge which is high valuable in the applicability requirement. Moreover, partial validation is possible. However, there are requirements that they are not taken into account such as incompleteness and ambiguity. The structure used is mainly for human use so the level of formality is relatively low. The strength of object oriented models is its distributed composition because its ability to handle new classes as well as new updated instances. Moreover, partial validation is also possible. In the TEA Project the quality information requirement is also fulfilled because of the description of the quality of the cues which is also useful to handle incompleteness and ambiguity requirement. However, the formality requirement is not completely fulfilled because of the encapsulation. Moreover, applicability to existing environments is fulfilled only with strong requirements which are often not given by ubiquitous computing systems. The strength of logic based context models are its distributed composition and level of formality. However, there are requirements that are not fulfilled such as partial validation which is difficult to maintain, quality of information, incompleteness and ambiguity or applicability to existing environments because of the difficulty of finding full logic reasoners in ubiquitous computing devices. Moreover, although this model uses an extremely high level of formality, without partial validation the specification of contextual knowledge is very error-prone. The strength of ontology models is its distributed composition, partial validation and level of formality. Depending on the approach the quality of information and ambiguity is also fulfilled (CONON support it but not CoBrA). Incompleteness is covered by all approaches. However, the applicability to existing environments depends of each approach. In CONON ontologies it is necessary to use integrate elements which handles OWL-DL for knowledge representation, but in CoBrA ontologies it is not necessary because of its broker-centric agent architecture. It can be seen that the ontology models are the models which covers best the majority of the requirements listed but it does not mean that the other approaches are not good enough for modeling context-aware applications [BCB+10; BDR07].

## 2.4 TOSCA

As it was said in the section 2.1, companies have started to introduce Cloud Computing in order to use virtualized computer storage and network resources and automate the orchestration of such resources reducing by the way costs of management and operation, which has a vital importance.

Enterprise systems are characterized usually by a high complexity and by consisting of a great amount of individual components, each one providing a different functionality, aggregated and orchestrated to offer a composite application. The type of architecture which mainly share this kind of applications is modular, so by the use of cloud technologies they benefit from properties such as elasticity, flexibility, scalability and high availability which reduce costs as the maintenance is outsourced.

One of the challenges of this kind of applications is the development effort in time, costs and the error-prone when done by hand. It is usually carried out in a manual way by using scripts or even by completely manual work. For that reason, portable applications which are automatic in deployment and management are highly desirable [BBKL14].

TOSCA (Topology and Orchestration Specification for Cloud Applications), an OASIS standard, has introduced a way to model cloud application and to manage them in a portable way.

In the Figure 2.7 it can be seen how TOSCA describe a service template.

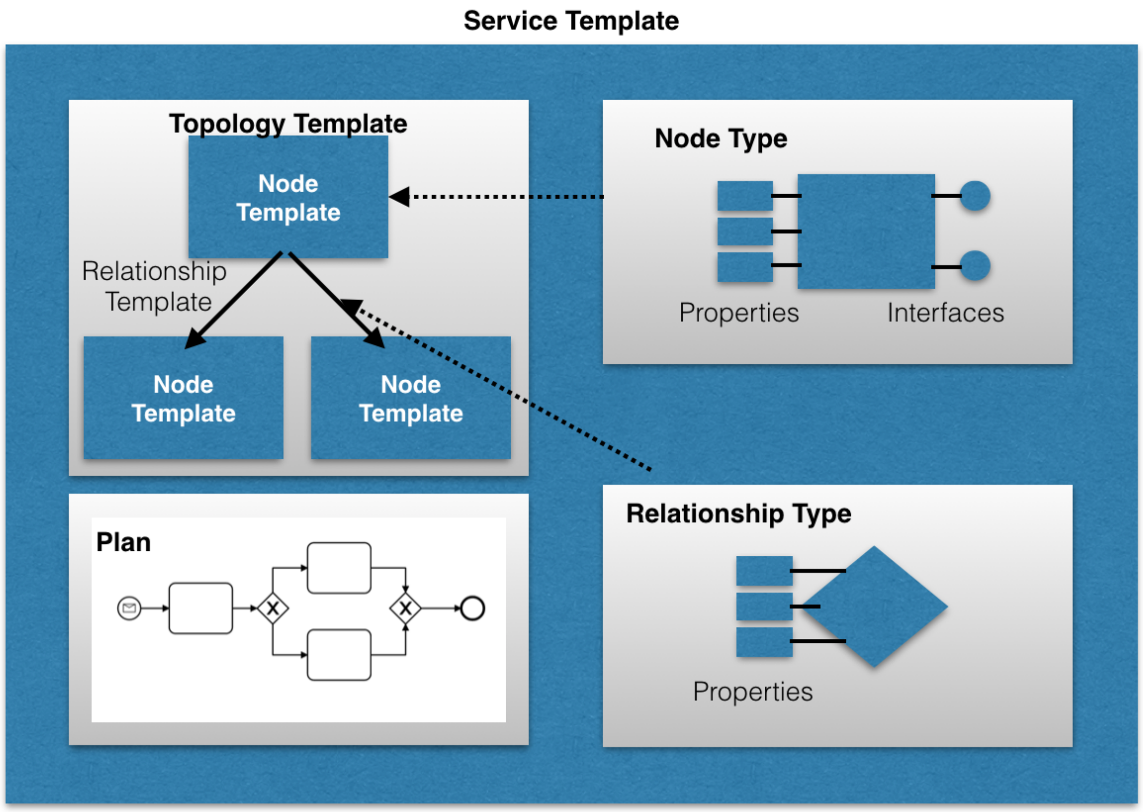


**Figure 2.7:** Overview service template topology by TOSCA

TOSCA separates application structure into so-called Topology Templates which are graphs composed of the application components, represented as node templates, which are semantically interconnected by relationship templates.The behavior of TOSCA applications is specified in plans. As it is shown in Figure 2.8, TOSCA is based on typing. Each node template and each relationship is related to a specific type, introducing by this way more valuable information to the topology. By introducing the typing, it is possible to define life-cycle states, policies associated, artifacts and management operations.

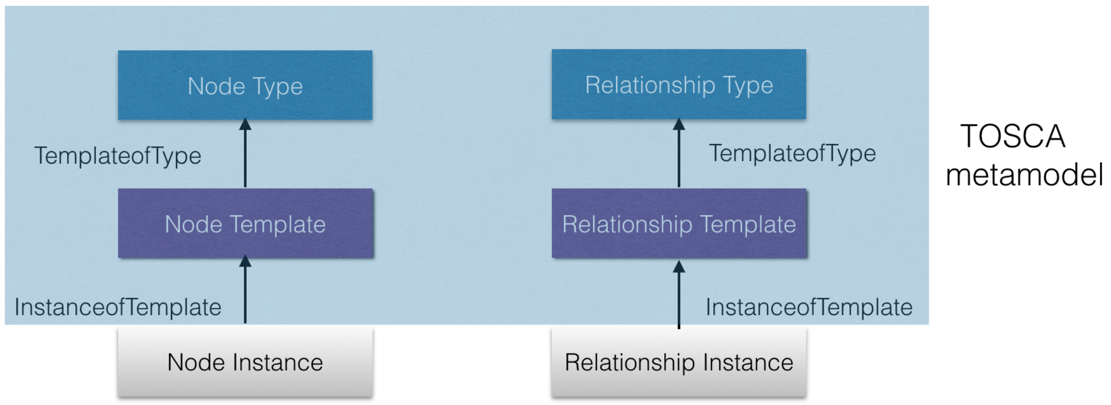


**Figure 2.8:** Conceptual Layers of TOSCA-based applications

The TOSCA meta-model can be compared to abstract classes of Programming Languages. There are abstract classes which define the functions that each node or relation could execute. This is the first layer of the TOSCA meta-model, called Type Layer. Secondly, there are concrete classes which extend to the abstract ones in which represent how it is going to be instantiate the different nodes and relationships, known as Template Layer. Finally, the last layer, known as Instances, represents the real instances of the components and relationships, which depend on the TOSCA runtime.

## 2.4.1 Topologies and plans

Although TOSCA plans can be implemented in any language, TOSCA proposes the use of languages such as BPMN 2.0 because of the standardized graphical rendering that it offers, not enforcing to make cyclic graphs. Moreover, it also offers a well-defined execution semantics. The main difference between other workflow languages, such as BPEL, is that data objects are used to model meaning which offer a high granularity in contrast with object modeling. To determine what the workflow functions are and in which order they are carried out, BPMN needs to determine tasks, which are used for example to call services, to execute scripts or to trigger human actions; and the control flow between them.

TOSCA specification defines Build and Termination Plans. The so-called Build Plans are in charge of the deployment and installation of a service. These plans are characterized by the workflow shown in the Figure 2.9

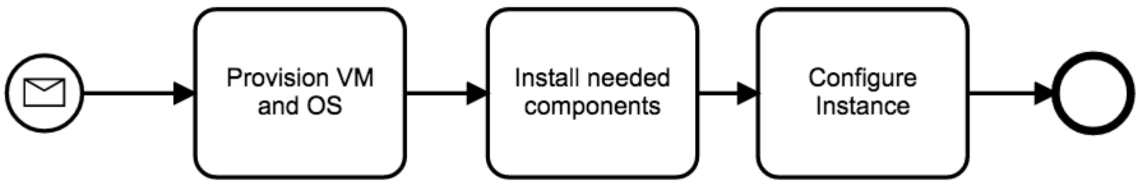


**Figure 2.9:** Example of a simple Build Plan

First of all it is needed to install and configure resources in which the service we want to integrate to the system is going to be deployed. These resources can be installed before in the system and started when they are required or in contrast they could be cloud resources which are requested to cloud servers, offering a high flexibility of the adaptation of the system depending of the actual requirements. When they are started, then the service is installed. Depending of the customization of the service, it could be a simple execution of a script given by developers such as installation of databases or web servers, or more complex processes. Finally in the last step the configuration of the instance is done. In this step, it is introduced values of the properties of the Node and Relationships templates of the Topology which are used as configurable parameters.

In contrast, Termination Plans orquestrate the termination of the service. In this plan, first of all the service would be uninstalled in the topology and finally it would stop the service of the resources in which the service was deployed in.

A compliant management platform is required to execute the service templates in TOSCA. This is known as a “TOSCA container”. This runtime guarantees that all the implementation artifacts needed for the management operations are available before the service is instantaited. Moreover, management service templates, access to the topology model and deployment artifacts handling is also is given by these runtimes [BBKL14].

Although all the benefits that this standard has introduced, the complexity of the modeling and management plans has become a challenge to developers. Oliver Kopp et.al has introduced a new domain language called BPMN4TOSCA [KBBL12] in which the complexity is reduced by the provision of ways to access properties independently of the modeling and dynamic plan to access to topologies. Moreover, it enforces the strong integration of management plans and operations. Scripts also take an important role in the topologies so the objective of this language is that scripts on nodes can be easily executed.

### 2.4.2 Declarative & Imperative Management of Cloud Applications

One of the objectives of developers is to reduce costs and timing in the way they implement a service. Traditionally, the way to design and develop a service was to define all possible situations that may occur and the performance of the service in those situations. In contrast, an approach in which the management logic is more important instead of the definition of plans [SXV].

- **Imperative processing**: This approach allows a precise specification of all management logic enabling a high level of customization in the application provisioning and creating an explicit process model which can be executed automatically by a runtime. By using this model, it is necessary to define the control flow of activities, the data flow between them and finally all technical execution details. In TOSCA, it is required that all needed management logic is contained in CSAR (Cloud Service Archive). These archives contain fully automatically executable Management Plans which describe management tasks such as provisioning, scaling or updating provided by their components or by publicly accessible services (Amazon Web Services API). Therefore, not only does it describe what has to be done, but also how the whole provisioning tasks have to be executed. However, there is needed a huge amount of specified statements in order to execute the application because of the lack of logic of the runtime itself. It makes a labour-intensive nature, time consuming and error-prone because of the need of orchestrating heterogeneous management services. Moreover, the need of creating the application's plans from scratch, in complex topologies such as Cloud Application topologies, these manual creations are easy to develop and error-prone.
- **Declarative processing**: This approach, in contrast with imperative processing, focuses its effort in describing management logic, not in defining plans. Therefore, it is needed a precise definition of the semantics of the components of the topology. So, it is defined, by this model, what functionalities have the applications, but not how they should be technically implemented. There are declarative technologies, such as Chef or Puppet, whose focus is describing the desired state of the components without specification of tasks that should be done to achieve that state. In TOSCA, TOSCA Runtime Environments interpret application topologies by introducing management logic without the need for plans, requiring an accurate definition of the semantics of nodes and relations between them. This approach solves the difficulties of imperative processing in modeling complex topologies

and makes the process easier and faster. However, this processing produces a lack of flexibility in provisioning capabilities and pre-defined operations during the runtime because TOSCA runtime Environments have to understand the components or at least the management operations they provide. In addition, the declarative approach is only suited for common applications without any level of customization. Nevertheless, this approach supports native Cloud Providers, allows to use reusable artifacts and simplifies the applications deployments. [EBF+17]

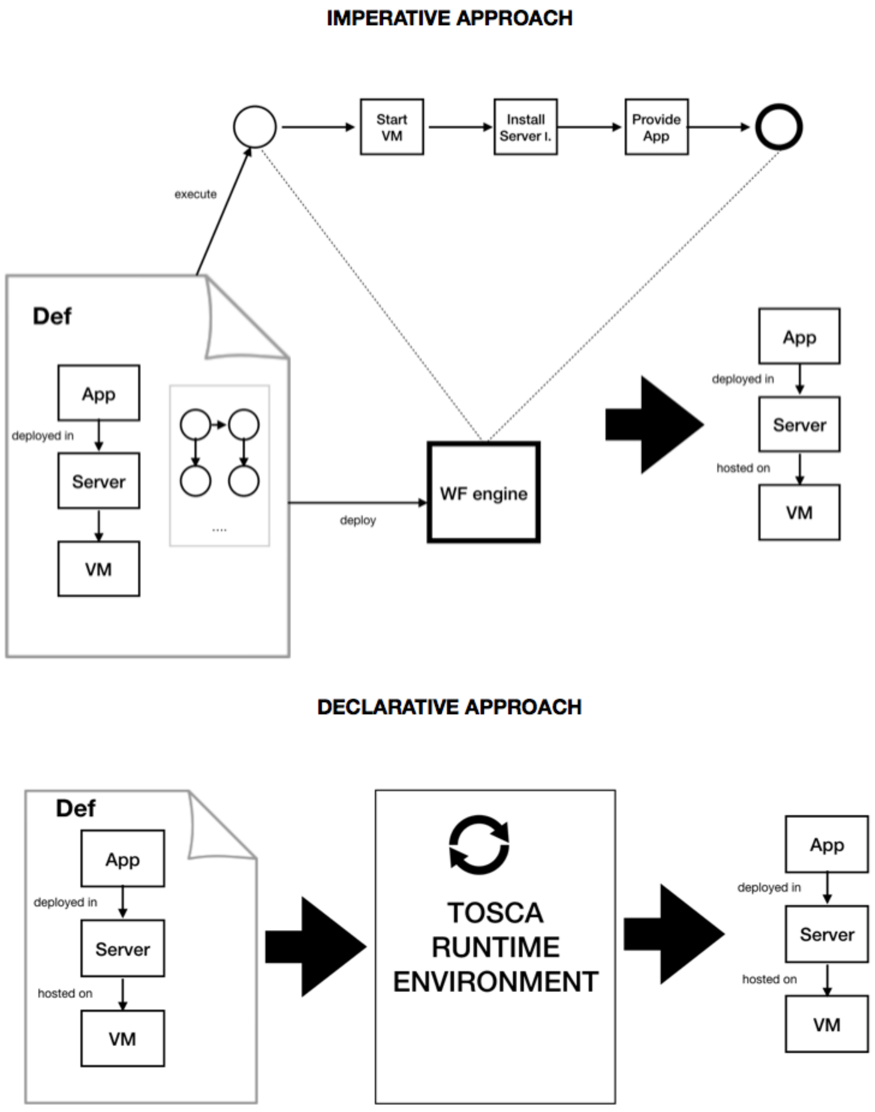
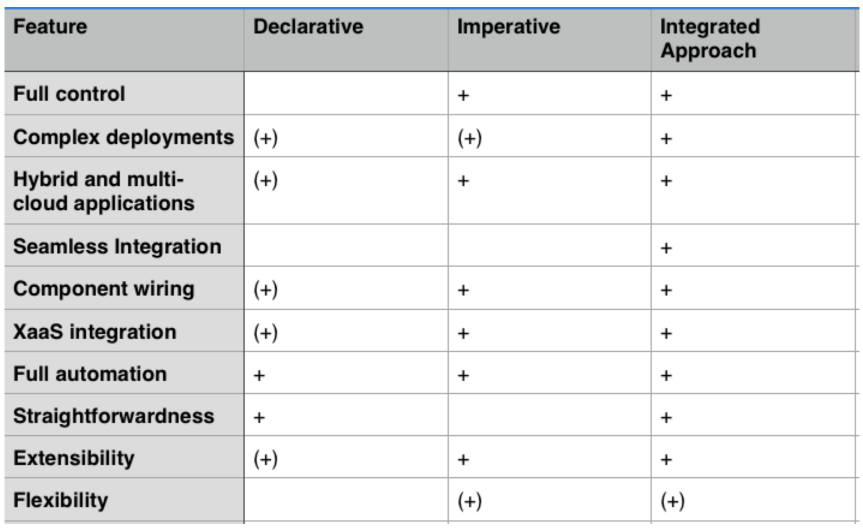


**Figure 2.10:** Imperative and Declarative Approaches

In the Figure 2.10 the basis of both approaches is represented. In the imperative approach, a Workflow Engine manages business processes and which automates the execution of the appropriate tasks in a fixed order required in the different situations that can be expected during runtime. In the left side of the first figure, it can be seen an abstract workflow whose aim is to install an application in a server instance: first of all a virtual machine (VM) is started. Then an instance of the server which is going to provide the service is installed and finally the service is installed and started. The execution of this workflow would derive to the final topology represented in the left side. On the other hand, by using the declarative approach it is only necessary to define the different node and relationship templates which can be integrated in the topology and then during runtime the TOSCA Runtime Environment would be the responsible of deploying the final topology.

To sum up, in the Table 2.2, it can be seen a summary of properties of each approach. It can be seen that if an integration of both approaches is made, all the requirements of an automated application provisioning can be handled.



| Feature | Declarative | Imperative | Integrated Approach |
|---|---|---|---|
| Full control |  | + | + |
| Complex deployments | (+) | (+) | + |
| Hybrid and multi-cloud applications | (+) | + | + |
| Seamless Integration |  |  | + |
| Component wiring | (+) | + | + |
| XaaS integration | (+) | + | + |
| Full automation | + | + | + |
| Straightforwardness | + |  | + |
| Extensibility | (+) | + | + |
| Flexibility |  | (+) | (+) |

**Table 2.2:** Evaluation of approaches [BBK+15a]

# 3 Related Works

In this chapter we show related works that have been taken into account in this thesis.

## 3.1 Integration of Situation-Awareness in Workflow Models: SitOPT

Integrating context in workflow modeling is not an easy challenge. Workflows are defined as a set of activities which are interconnected in order to achieve a business objective by executing this set in a runtime environment. So in order to take into account changes in the model during runtime, it was necessary to introduce into the workflow model all the possible situations that may occur and changes must be monitored and acted upon by the system. The SitOPT project has presented a solution defined as *ProSit Method* [KBS+16] which enables to model traditional workflow models and automatically adapts and transforms them to situation-aware workflow models. The system developed (*SaWMS* or *Situation-aware Workflow Management System* has the capability of taking advance of low-level sensor data in high-level situations in order to adapt workflows dynamically. In order to process the context data and the detection of which situation is happening, the SitOPT architecture is used which is mainly structured in three layers: Sensing Layer, Situation Recognition Layer and Situation-Aware Workflow Layer. The first layer, Sensing Layer, is composed by all the low-level sensors , measure data associated to the context and propagate it to the upper layers. The second layer, Situation Recognition, is in charge of processing the data taken in the first layer using different techniques such as filtering or aggregating. The last layer, Situation-Aware Workflow Layer maps this aggregated data to high-level situations.

The adaptation of the workflows, as said before, is handled by the SaWMS and its Situation Handler component. The SaWMS system is in charge of executing the workflows and send service requests to the Situation Handler. The Situation Handler is a situation-aware bus which selects the most appropriate workflow fragment capable of executing the requested operation invocated by the SaWMS.

In order to transform traditional workflows into situation-aware ones the method, previously presented, Prosit is used. This method is mainly based on six steps. First of all it is necessary to model the workflow desired. In the second step, a set of fragments of a situation-aware workflow are matched against the workflow model in order to detect subworkflow models. The fragments are aware of not only the action they are carrying out but also the situation in which this fragment is executed. After this detection, it is necessary to check if some overlap between fragments exists. In that case, it is manually selected which fragment should be executed. Moreover, if the data flow of the original and new workflow do not uniquely match, it is configured. After this adaptation,all the matched fragments are replaced from the original workflow to single placeholder activities, which include each one a definition of the invocation to Situation Handler, transforming it to a situation-aware workflow. As the final step of the method, this model is provisioned.

This solution was evaluated in a temperature regulation mechanism case study. In order to do that, first of all there were modeled as fragments the main actions of the service. Secondly the workflow introducing all the tasks necessary to regulate the temperature was modeled. Afterwards, the transformation to a situation-aware workflows was carried out by the substitution of the logic to situation-aware placeholder activities. When the execution takes place, and a variation of temperature is detected an instance of the workflow invoke the activity required (reduce temperature) in the Situation Handler which by the use of the SitOPT system knows the temperature of outdoor area and analyze which is the best operation to carry out in the situation (open the door if the temperature of outdoors is lower or regulate the temperature by the climate control...)

## 3.2 Contextualization of Business Process

One of the big challenges in Context-Aware Systems is to determine which context variables have impact in the process and how different values of these can affect the structure and behaviour of applciations. Michael Rosemann et. al [RRF08] in their research have focused in the conceptualization of context in business models and introduce a meta-model in order to integrate them as a whole. They consider that the context variables that should be taken into account have impact in the whole process, not in only one step of the workflow. It is necessary to have in mind the goals of the process because they determine strategic, operational, regulatory relationships between the steps of the process. This method is really promising because the context factors which have importance in the process can be predetermined and modeled and by the integration of the context variables with the goals, the flexibility to handle changes in the environment can be modeled.

In the meta-data proposed by this research, the context is considered as a layer framework composed mainly by four disjoint categories. The first one, known as Immediate Context, include standard process description such as control flow (order of steps), data (information required in the execution), application (what application supports a determined step?) , resources (what capabilities are needed?) and so on. These elements are fundamental for the comprehension and execution of a business process. To execute a business process, if no contextual changes are detected, these elements are enough. The second layer, known as Internal Context, which encompasses the internal environment of the organization relevant in the process such as resources, strategy, values, culture, policies... This context is so valuable because a change in a policy or in a strategy have a big impact on the business process. The third layer, known as External Context, captures elements that are not part of the organization but may affect the business process such as suppliers and regulations . Finally, the last layer known as Environmental Context covers the environment in which the process is executed such as weather, time and workforce related factors. These variables may change regularly so the changes related to elements of this layer are the most frequent.

With the use of this meta-data it was evaluated in this research how context could be analyzed and select the variables more relevant to the process. First of all, it was necessary to identify the hard-goals and soft-goals of the business and their suitable variables. Secondly, decomposition of the process was needed depending of the goal-relevant information. Afterwards, it was required to evaluate the relevance of the context variables in the goal associated. Then, the identification of the contextual elements is done and finally these elements were catalogued with the use of the meta-model previously described.

With this procedure, a case study was done related to the check-in process of a major Australian airline. The process usually works in a regular environment but there are situations that may require that the processes change. For example, weather conditions (bad weather conditions) or time conditions (holiday season) or server failures may provoke that customers miss deadlines because of big queues in the check-in, so changes in the structure are required such as the introduction of more check-in counters, using the counters of priority clients to all the economic passengers too or the reduction of the timing of check-ins by eliminating special requests. By using the procedure, first of all they selected a goal that was to "Minimize throughput time". Then the decomposition of sub-processes and sub-goals was done. To determine the relevance of context used single functions which eased the determination and to identify contextual events, for example by analyzing which elements of the context were not inside the immediate context such as elements that affect the availability of internet connection or alarms. By applying the categorization and classification of the context, eases the anticipation of important external triggers and the adaptation to the change are possible.

## 3.3 Integration of Imperative and Declarative Approach to model Cloud Applications

Cloud applications are getting more and more complex, using heterogeneous Cloud Providers which allow to employ services on different levels (IaaS, SaaS, PaaS), and it is necessary to be combined and integrated to get a fully automated deployment and management process. In order to do that, it is highly valuable to integrate both imperative and declarative approaches profiting from their benefits.

By using declarative processing in standard but complex activities, such as, configuration of a certain application component (installation of virtual machines, installation and configuration of an Apache Web Server. . . ) and using imperative processing for more customized activities, an efficient of application provisioning is achieved. [BBK+15b]

Uwe Breitenbücher et. al [BBK+14] have introduced a solution to combine the two approaches in order to generate fully automatic provisioning plans which are workflows that can be customized by developers after generation. The Provisioning Plan proposed is composed by three steps. In the Figure 3.1 the Provisioning Plan Concept is represented.

First of all, the Topology Template is evaluated and it is transformed to a Workflow Provisioning Order Graph, which indicates the order of provisioning of all the components that integrate the topology and their relationships. Secondly a Provisioning Plan Skeleton is generated which defines the structure of the final Provisioning Plan using a specific workflow language such as BPMN or BPEL. Finally, it introduced executable capabilities which transforms the skeleton in a fully automatically executable workflow. This workflow is executed in a TOSCA Run-time Environment, so-called *OpenTOSCA* which offer fully automated provisioning with the presented concept.

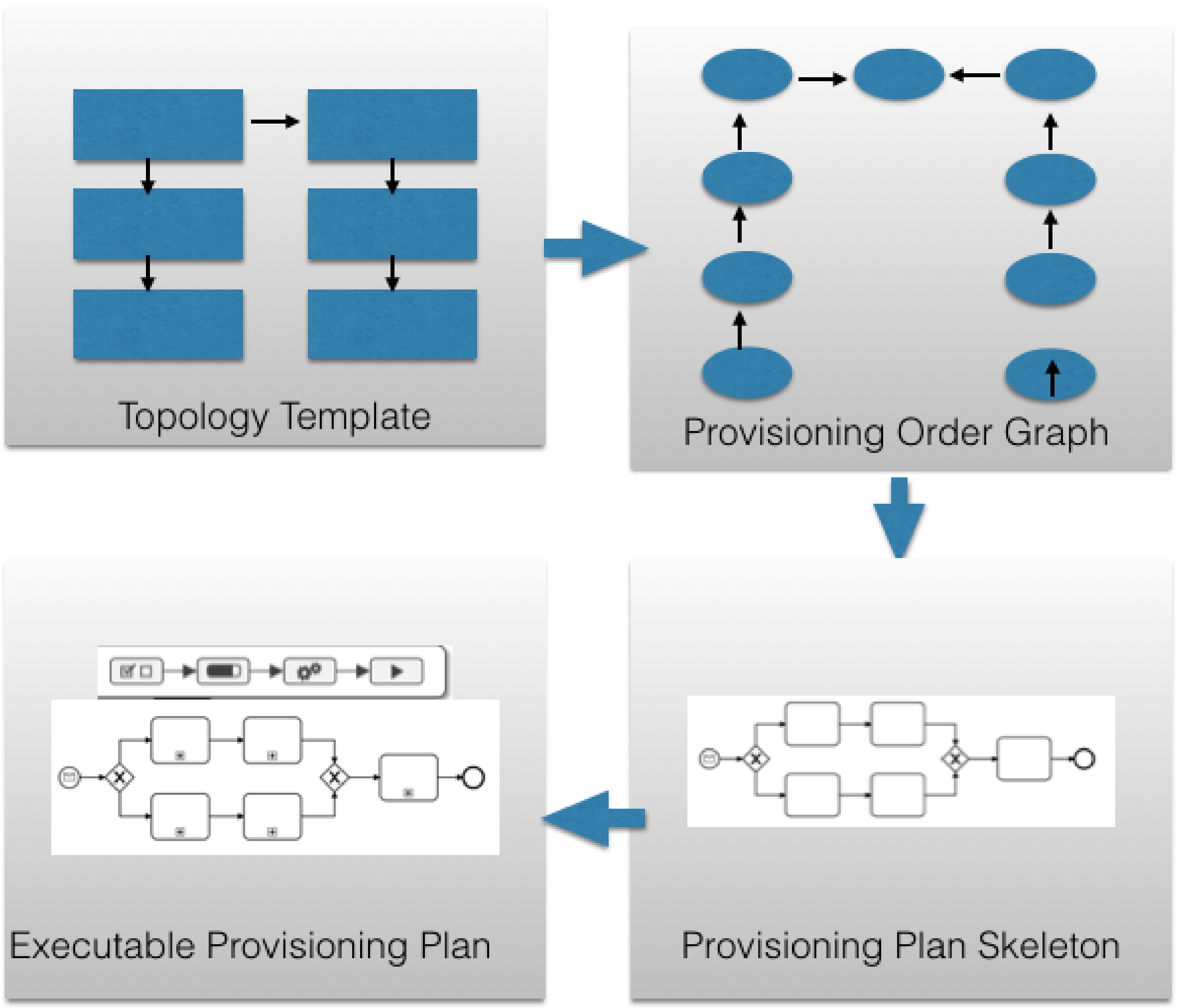


**Figure 3.1:** Three steps of Generating Provisioning Plans [BBK+14]

# 4 Motivating Scenario

In this chapter we present a scenario which we use to present and evaluate the modeling solution proposed. Moreover, we describe some use cases of this scenario which use context data to provide different services. Finally, we specify and present the formal requirements to model this scenario.

## 4.1 Description

Because of the incorporation of mobile devices in a growing number of systems, it is needed to take changes of location and temporal connections between different systems into account. By this location-awareness, a user can access to services available in the area they are or the systems can take advance of the facilities that are available to improve the performance of the services offered.

The Figure 4.1 represents our scenario. There are two systems, each one has its services which are deployed by the different components interconnected, and one user device, that depending on its location, is temporally connected with the system closer and more suitable to it. Moreover, additional components which can be portable and could introduce new services in a system can be temporally connected to the required entity. Depending of the provided service to the user it may be necessary to connect to specific components of the system depending on the performance and efficiency of the components of the overall application.

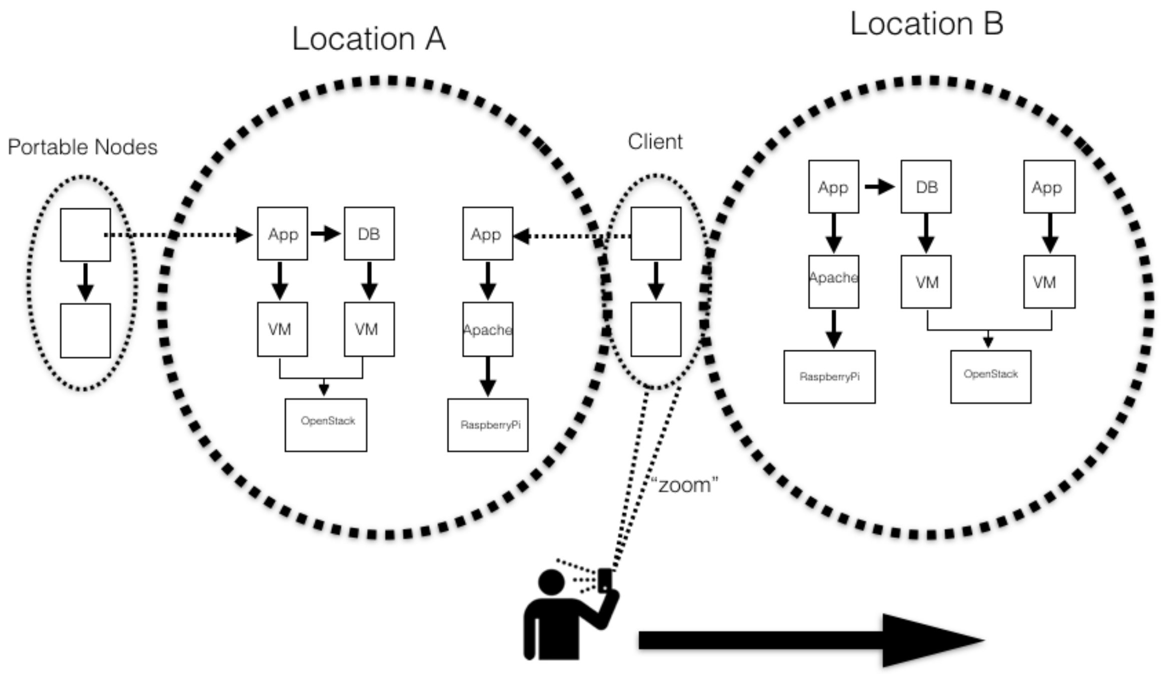


**Figure 4.1:** General Description of Motivating Scenario

This scenario encompasses different use cases that not only take spatial context into account but also other context variables, which were described in the section 2.3. Associated to the different systems like temporal connections to external systems such as back-ends servers, connections to emergency services because of detection of extraordinary environments situations, or connection to better performance external entities detected are also part of the context. In the following list we present potential use cases of this scenario that try to cover most of the possible usages of context in the system:

1. **Connection to server instances**:In this use case, we consider a user that moves from location A to B, as it is shown in the Figure 4.2, which is using a service provided by the node template called SA. In order to implement this use case it would be necessary that the node template is available as an instance in the topology close to the location of the user and to have resources in which it can be installed in order to connect to the user. In the 4.2 we specify that an instance of the server SA is hosted on the VMS A or VMS B depending of if the user is close to Location A or B respectively.

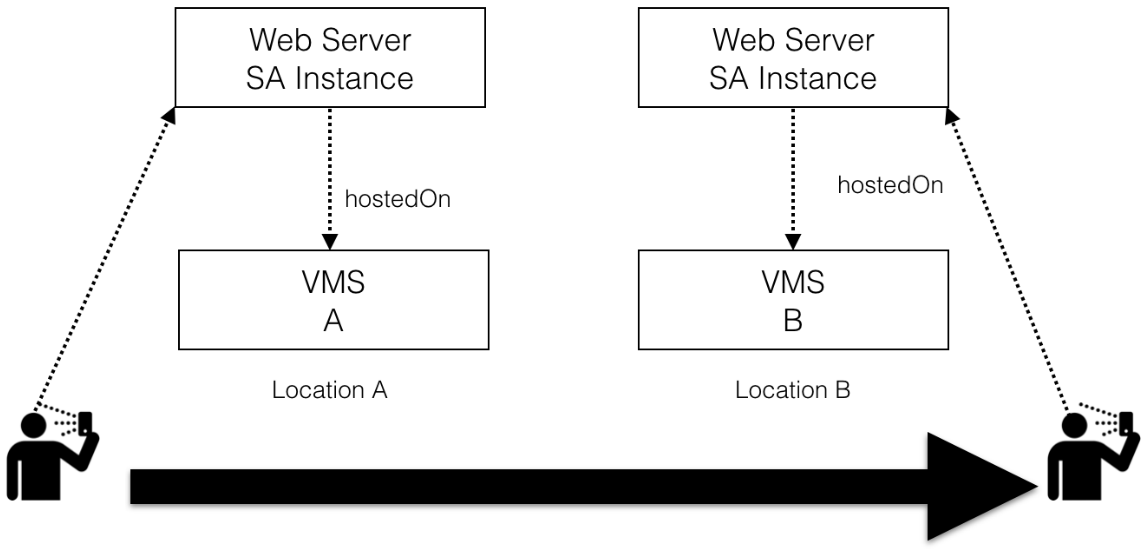


**Figure 4.2:** Representation of the first Use Case

A variation of this use case is to include services that depending of the location of the user, are provided by different service templates. As well as the first one, it considers an user that moves from location A to B and it is connected to the node template called SA or SB respectively. In order to implement this, it would be required that an instance of the node template associated to the location where the user is installed in available resources of the topology. This means the system can allocate the resources dynamically when it detects the user.

2. **Temporal Connection to Environment Sensors**: A context-aware system is characterized by having a dynamic topology in which the relationships between components are not fixed and the incorporation of new components is possible. In this use case, (see Figure 4.3) are analyzed changes of time frames. It considers a Node Instance that is a central component of a system which depending of the period of time (day/night) is connected with the gateway of light-detector sensors of outdoor or indoor areas respectively in order to take valuable environment measures. For that reason, an establishment of a connection with the convenient gateway and a disconnection with the previous gateway would be required.

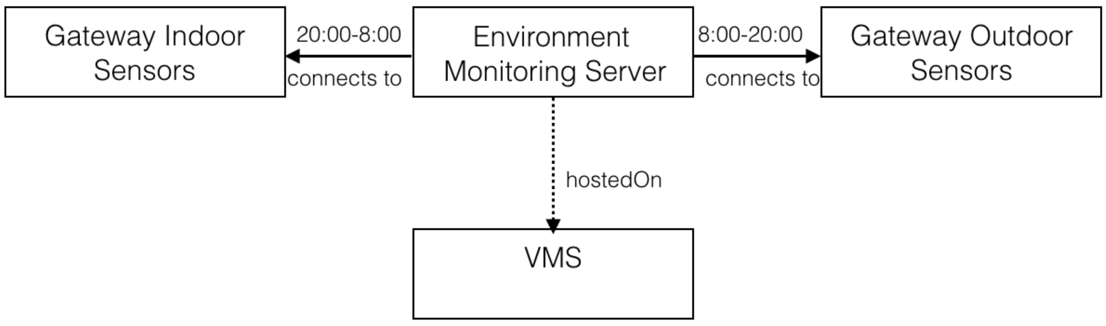


**Figure 4.3:** Representation of the Use Case

3. **Detection of Points of Failure**: One of the benefits of being aware of the context is that detection of failures inside or outside the system is faster and adaptations to decrease the impact is possible. In this use case, (see Figure 4.4) a node called NA is connected to a Server Instance, called SB. If a time threshold is exceed, it is evaluated what available instance of server is more suitable and after disconnecting with SB it is connected with this suitable instance, called SC.

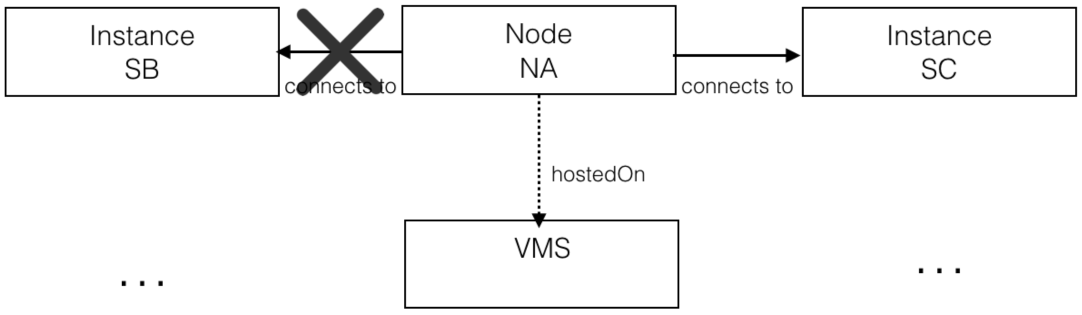


**Figure 4.4:** Representation of the third Use Case

As an alternative to this use case would be that instead of monitoring the availability and performance of a server connected, analyzing the performance of the node itself and if the execution is not as it was expected, finding resources in which host the instance and improve the performance, as shown in the Figure 4.5. In this alternative the type of context used would be device context, this means, the information taken would be what it is offered by the device itself.

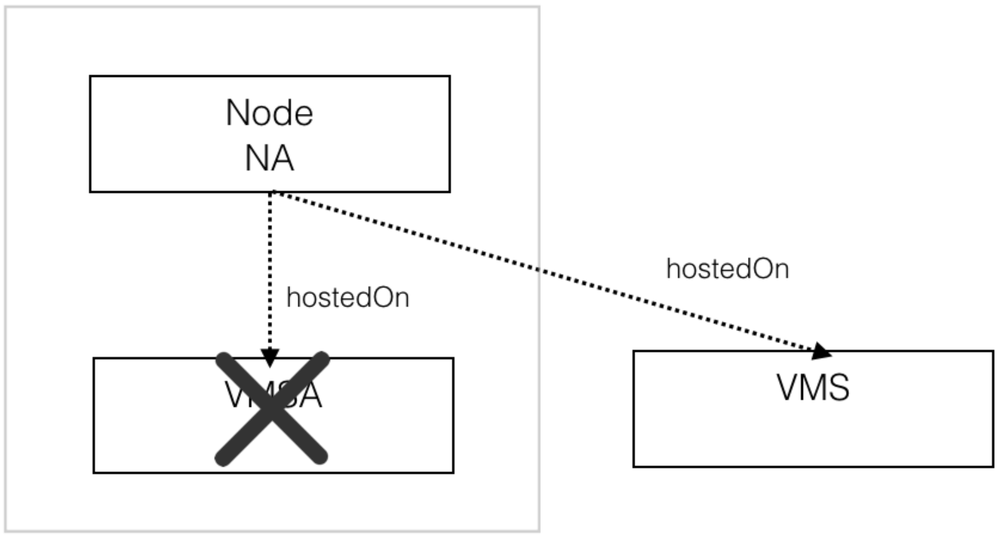


**Figure 4.5:** Detection bad performance

4. **Video Streaming**: Not only the context-aware detection is about the performance of the system, but also, about new components that can improve the quality or introduce new functionalities into the system. In this use case, (see in Figure 4.6) it is studied how a video provider, which sends streaming in a traditional way, when it is detecting a good WIFI connection and nodes which could improve the quality of streaming by using Internet transmission, establish a connection with it and changes the way of stream broadcasting.

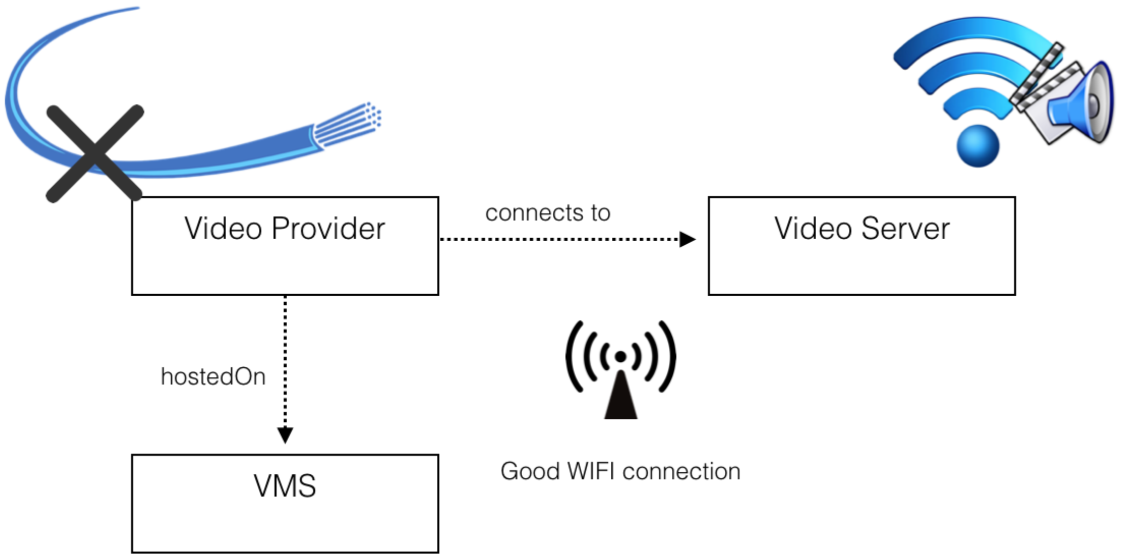


**Figure 4.6:** Representation of Forth Use Case

5. **Emergency Situation**: Environment context is as valuable as space or time context. Emergency situations are one of the areas in which context-aware systems are beneficial to the most because a quickly reaction is highly valuable in such situations. For that reason, this use case (see in Figure 4.7) analyze how when an emergency situation is detected by the monitoring entity, a connection to the emergency server which can handle this situation by activating alarms or handling the protocol to act could be established.

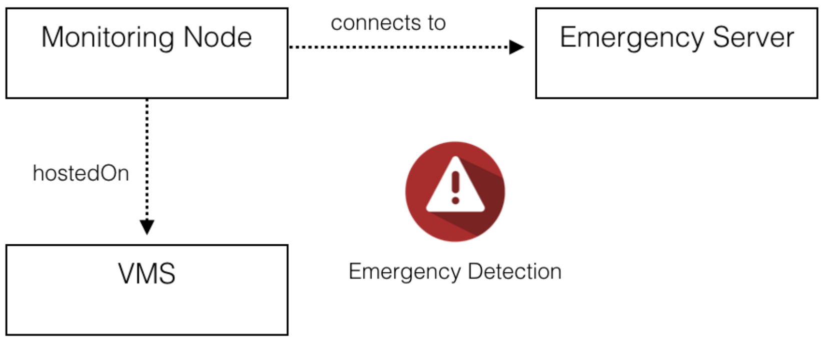


**Figure 4.7:** Representation of Fifth Use Case

## 4.2 Modeling requirements

In order to model this scenario we identified the following requirements that have to be taken into account:

1. **Active Context Monitoring**: Context is constantly changing so it is necessary to adapt as quickly as possible. This means that it is needed that all the processes carried out in the system should be aware of situation shifts and if it is necessary, executing procedures to adapt to them. Active Monitoring is valuable in emergency situations handling because when a emergency situation is detected, it is required to react to it as fast as possible.

2. **Monitoring System Execution**: It is needed to evaluate at runtime if the procedures executed by the different node instances of the topology is adequate to the context situation and moreover, if there are failures in the different process executed, try to solve them. For example, in the Detection of Points of Failure Use Case, Monitoring Execution is required to check if the performance of the node instances installed or the communication with external nodes is as expected, or if not try to find new resources which can improve the performance of the services.

3. **Context Variable Relevance**: Because of the amount of context variables which can influence an entire system, it is necessary to evaluate the relevance of each one in each process in order to know when it is necessary to react to situation changes. For example in the Connection to Server Instance Use Case the spatial context would influence most to the system.

4. **Portability of Node Templates**: As it was said before, there are services which need specific nodes templates and it is fundamental that instances of these node templates can be installed and started in the location where the users are. This is needed in the first use case proposed, because independent of the location, an instance of the server always must be available.

5. **Compensation critical procedures**: There are activities such as installation of node instances, establishment of connections that are vital to deliver the functionalities required. Errors in these procedures could provoke singles points of failure and they need to be compensate in case of any type of error during their execution.

6. **Unbuilding previous situation-related topologies**: When it is detected that the context situation has shift, before introducing the changes related to this adaptation it is needed to reestablish the previous modifications of the topology such as the installation of new nodes or establishment of new connections. It is required in all the use cases proposed.

7. **Events Relevance**: A system could have a big amount of events than can trigger different procedures and they may be executed simultaneously such as the change of location and time period or emergency situation and detection of changes in the network context. Some events have more importance than others so it is necessary to define a order in which the different events that could happen in the system should be handled first. For example, emergency events should be prioritized from other events because of the requirement of timely reaction.

# 5 Concept

In this chapter we introduce our conceptual modeling approach to the scenario described in Chapter 4. We evaluate the suitability of using Imperative or Declarative Approach in the use cases we presented and how the solutions proposed could be integrated in the TOSCA specification by modeling an architecture solution which is event-aware.

## 5.1 Concepts Context Modeling

Our first aim when designing the modeling solution was to provide an easy integration of the context in the topology. By that, the context model used, was required to be simple and easy to integrate the relevant context categories to the scenario and use cases described in the Chapter 4. Two possible context models were analyzed: Key-Value Model and Mark-Up Model, described both in the Section 2.3.1. Both of them are characterized by its simplicity to model the context but one requirement for us was to maintain a hierarchy of each relevant context category to the scenario. It is only possible to fulfill this requirement by using a Mark-Up Model.

First of all after deciding the type of context model required, we have analyzed what context categories are relevant in the scenario and use cases.

- **Spatial Context**: This context information is fundamental for the scenario. In all the use cases described, the location varies from Location A to Location B. So the hierarchy of the Location Category would have two markup tags associated each one with one location and whose values would be true or false depending of either the user is in that location or not.
- **Temporal Context**: This context information used in the use cases selected depending of the time frame the entity is connected or disconnected to. In the use cases, we selected two time frames: day and night, so it would be required two markup tags each one related to one time frame. As well as the modeling of spatial context, the values of the markup tags would be a boolean value depending of the timing.

- **Device Context**: This context information is used to analyze the performance of the entities integrated in the topology and also the performance possible entities to connect to when there are errors in the topology. This structure would have as mark tags all the nodes which could be possible to integrate in the system and the value of this markups would be true or false depending of its performance.
- **Network Context**: This context information is associated to the network performance which the system has. This information is required to improve the capabilities of the system in case of having a good Internet Connection. This structure would have as markups all the possible connections that the system have and as a boolean it would be represented if the connection is good enough or not.

This categories need to be formally defined [McC93]. The context categories would be defined by the following equations and the context would be integrated by these context structures.

$$c^0 \equiv SpatialContext = (LocationA, LocationB) \tag{5.1}$$

$$c^1 \equiv TemporalContext = (dayframe, nightframe) \tag{5.2}$$

$$c^2 \equiv DeviceContext = (node0, node1, node2.....) \tag{5.3}$$

$$c^3 \equiv NetworkContext = (connection1, connection2, ...) \tag{5.4}$$

$$c^0, c^1, c^2, c^3 \epsilon C \tag{5.5}$$

## 5.2 Concepts Imperative Approach

All use cases presented in the section 4.1 have in common that they need to be notified when a change in their relevant context has happened. In order to do that, we have designed a monitoring entity which is in charge of constantly checking if it has detected a change and notifies the necessary entities. In the Figure 5.1 the workflow which describes the performance of this entity is described. First of all, this is a process which should be repeated during the runtime of the system. For that reason, the sub-process checks if there are changes in a loop. In this sub-process, it is sent to the context data acquired by the available sources, then by using a variety of algorithms it is detected if there is a valuable change in the context measures (this activity is concurrent because it evaluates each variable at the same time) and if there is concluded that a change has happened, it would notify to the affected entities.

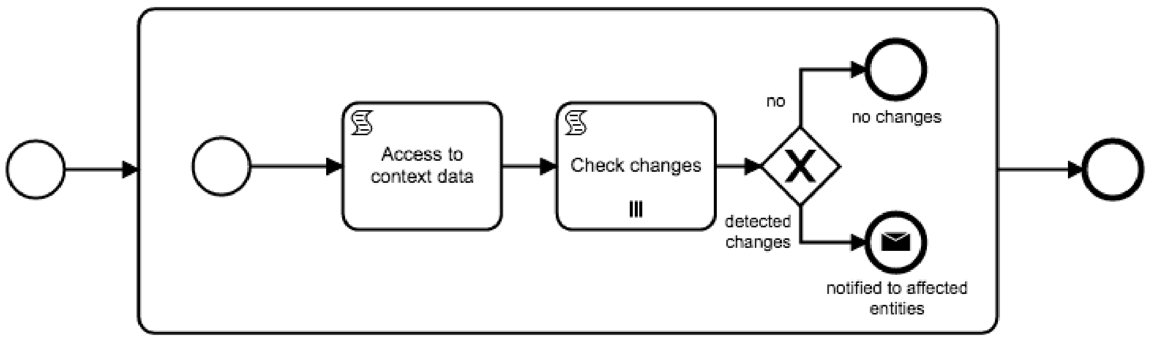


**Figure 5.1:** Monitoring Workflow

### 5.2.1 Imperative Modeling Solution of the 1st Use Case

First of all, the 1st Use Case called *Connection to server instances* was related to providing a service to a user not depending of the location.

To model this use case it is needed to analyze which events could happen during the execution of the system.

- Change of location: The monitoring entity would send when it detects a change of location a message to the system with the information of the new location. This event would trigger the adaptation of the system to the new location.
- Compensation events: Installation or start operations of different entities can provoke errors so it is necessary to handle them and compensate the changes in order to not affect the performance or the availability of these entities.
- Communication error events: There would be cases in which the connection between the device and the server could be denied. In this case, it would be notified to the central workflow which would handle the situation.

By taking into account the requirements presented in the section 4.2 and these events, in Figure 5.2 a model of this use case is presented. Modeling by using an Imperative Approach requires to define all the details of the workflow, (details, such as, what locations the user could be or what VMs are needed to be selected depending on the location) so to make easily understandable, it was needed to fix a low range of possible locations. For that reason, we selected two possible location in which the user can be: Location A and B. When a change of location is detected and the entity is notified by the monitoring workflow, a sub-process is executed in which first of all, fetches the

VMs suitable and after that all activities required to provide the service to the user. This sub-process is aware to the context, this means, that if during its execution a new change of location is detected, the execution of the sub-process is stopped. The changes introduced are compensated and the service is initiated again with the introduced changes. Moreover, if there are errors in the provide service phase, it would stop the execution of the process and would notify to the central workflow in order that the failure situation could be solved. When the provision of the service in the new location of the user is achieved, the system would be maintained until a new change of location is detected, which would trigger the sub-process previously explained or a termination signal which would trigger the end of the execution.

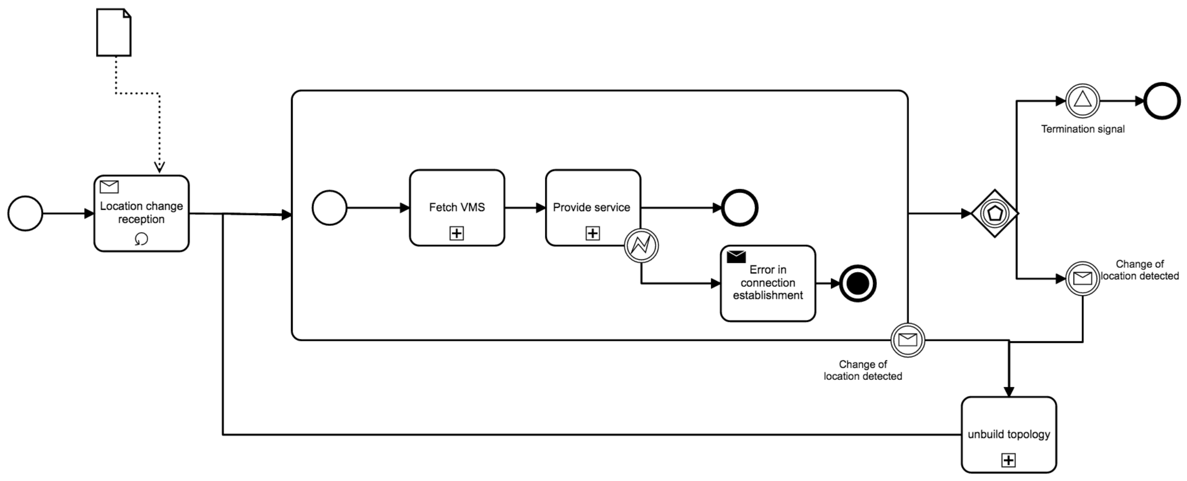

Location change reception
Fetch VMS
Provide service
Error in connection establishment
Termination signal
Change of location detected
Change of location detected
unbuild topology

**Figure 5.2:** Imperative General Model Approach of 1st Use Case

Two activities in the sub-process in the diagram are collapsed. One of them is the called "Fetch VMs activity". This activity is in charge of selecting the VMs resource related to the location of the user. In the Figure 5.3 the expanded description of this activity can be seen. When the VMs are fetched, a XOR operation is executed and depending on the location of the user is (A or B) it selects the VMs available in the system maintained in that location.

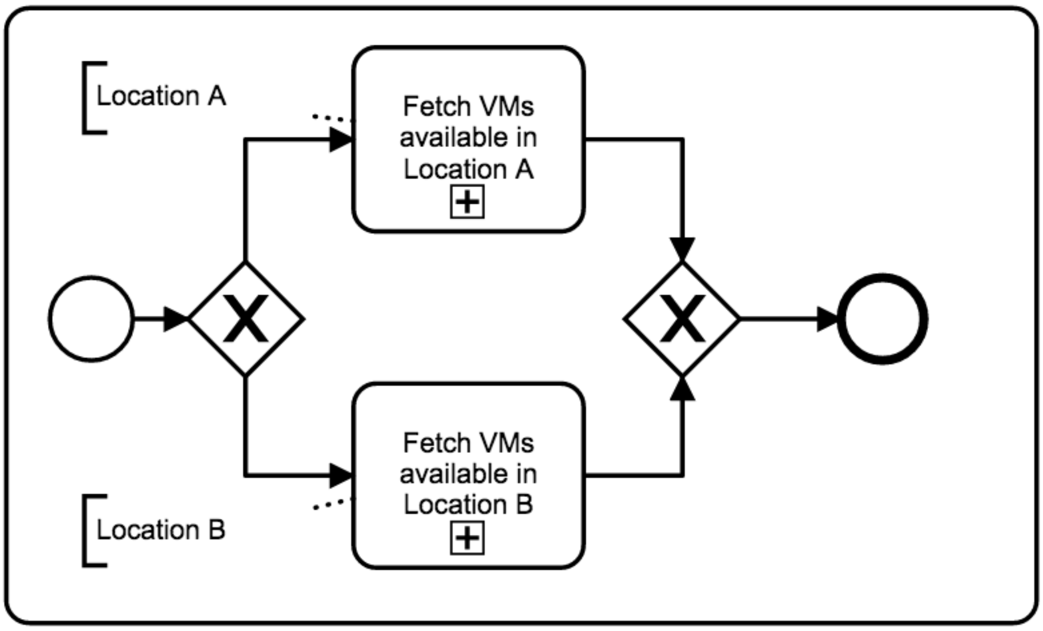

Location A
Fetch VMs available in Location A
Location B
Fetch VMs available in Location B

**Figure 5.3:** Expanded View Fetch VMs activity

The second activity is called "Provide Service". In the Figures 5.4 and 5.5 they are represented as two alternatives of implementation of this activity. The description of this activity differs depending of the type of the server instance required to provide the service. If the service is provided by a specific node template, it would be required to start the VMs previously fetched, then an installation of an instance of the fixed service template would be executed and finally the connection between the service instance and the user would be defined. On the other hand, if the service could be provided by different server templates an additional step would be necessary. The aim of this step is to select the available and more suitable service in which deploy the service.

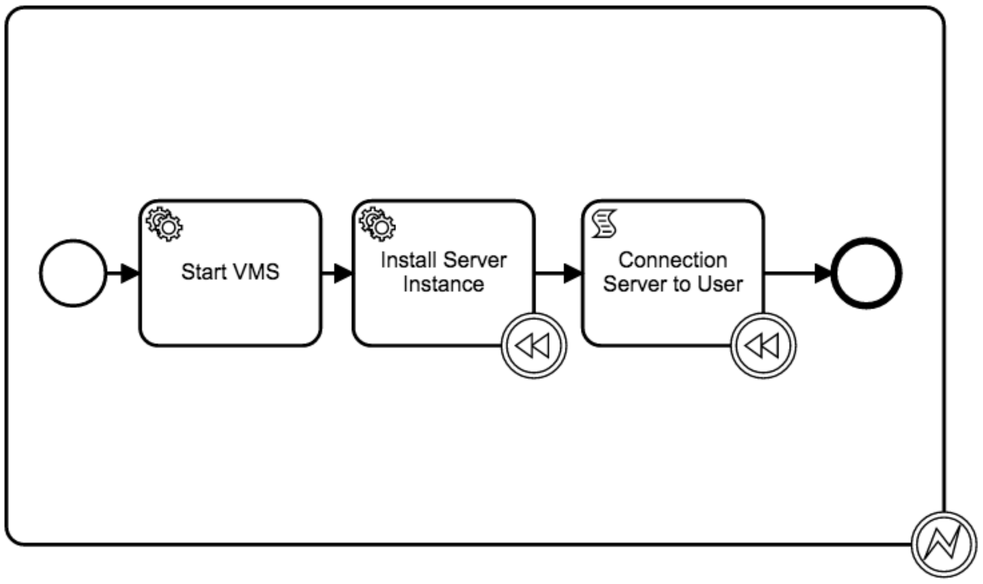


**Figure 5.4:** Expanded View Provide Service Activity - Fixed Node Template

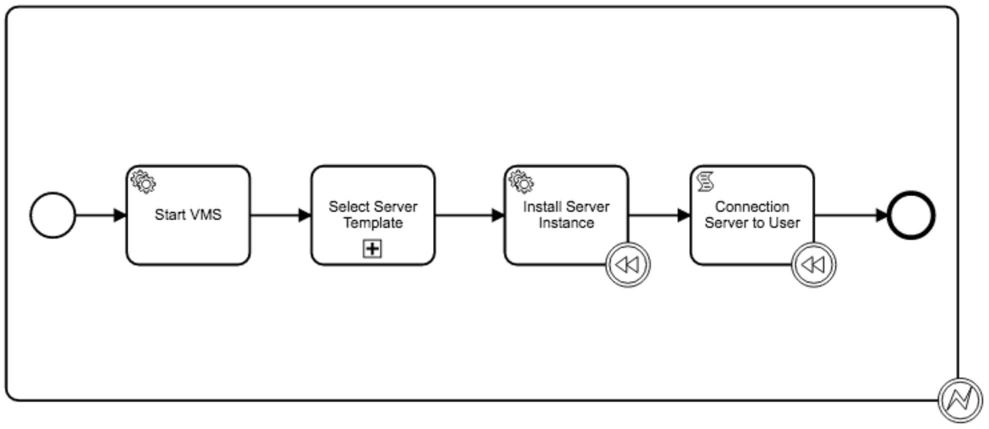


**Figure 5.5:** Expanded View Provide Service Activity - Not Fixed Node Template

### 5.2.2 Imperative Modeling Solution of the 2nd Use Case

The 2nd use case called *Temporal Connection to Environment Sensors* was related to establish temporal connections efficiently between nodes in the topology depending of the time frame of the day.

To model this use case it is needed before to analyze which events could happen during the execution of the system.

- Change of time frame: The monitoring entity would send when it detects that the frame day/night has changed. This would be the trigger to change the connection from one gateway to the other.
- Compensation events: Connection operations of different entities can provoke errors so it is necessary to handle them and compensate the changes in order to not affect the performance or the availability of these entities.
- Communication error events: There would be cases in which the connection between the device and the server could be denied. In this case, it would be notified to the central server which would handle the situation.

In the Figure 5.6 a workflow of a modeling solution for this use case is pictured. As well as the previous use case, we have considered only two gateways to which the environment monitoring node could connect because of the lack of flexibility that the imperative language approach has in run-time. In the solution it is represented that first it is necessary to start the monitoring node in order to handle the connections and receive the measures from one or the other gateway. Then it is initiated a sub-process that would be executed repeatedly until an error in the connections is detected. That would require an action of the central server. This sub-process waits until a change of time frame is detected. When it is detected the connection with the appropriate gateway is established and it is maintained until a new change of time frame is coming (a small threshold between time frames) in which the connection with the gateway would be disabled in order to start the sub-process again with the new temporal situation.

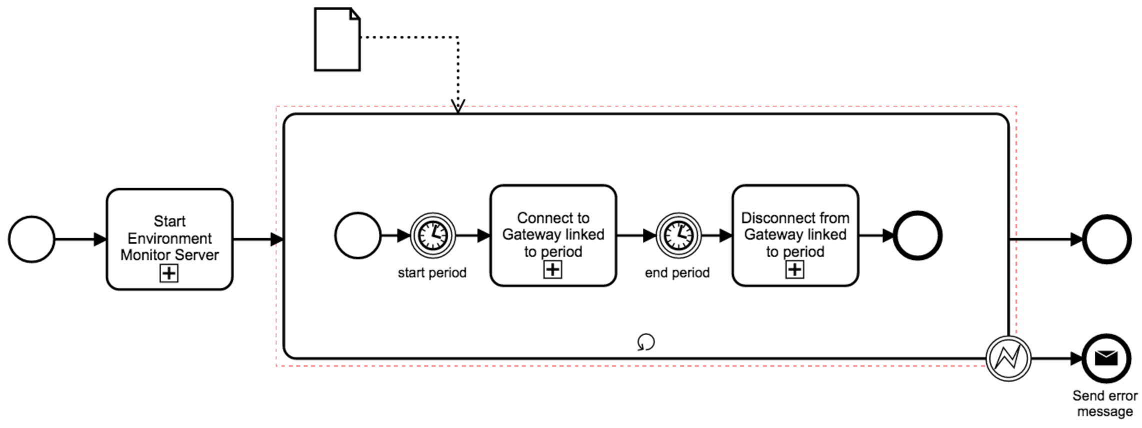

Start Environment Monitor Server
start period
Connect to Gateway linked to period
end period
Disconnect from Gateway linked to period
Send error message

**Figure 5.6:** Imperative General Model Approach of 2st Use Case

Two activities represented in the sub-process are collapsed: the connection and disconnection activities. In the Figure 5.7 we represent the expanded version of these activities.

In this expanded version it can be seen how the selection of the gateway to connect to is defined. If the time frame detected is the night the gateway to connect would be which is in charge of the sensors of indoor and in the other case would be the outdoor one.

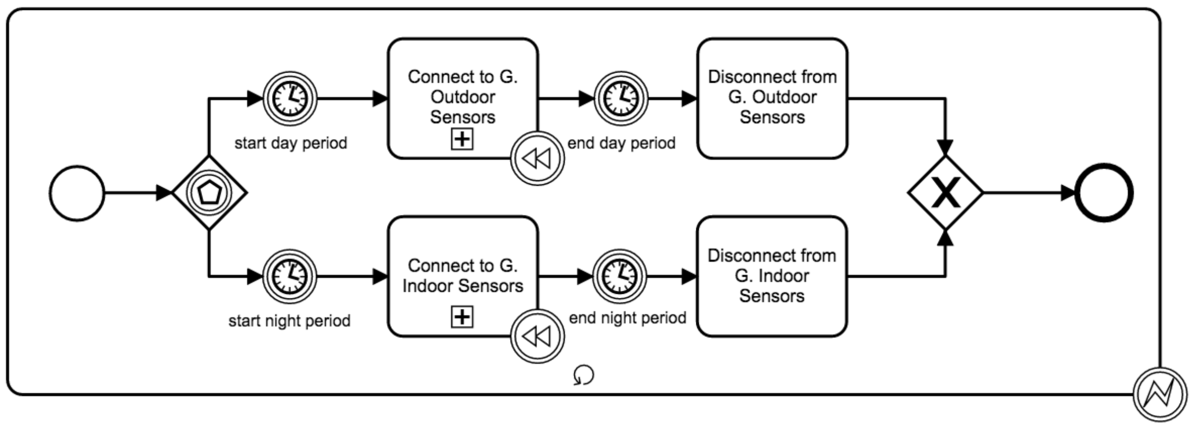

start day period
Connect to G. Outdoor Sensors
end day period
Disconnect from G. Outdoor Sensors
start night period
Connect to G. Indoor Sensors
end night period
Disconnect from G. Indoor Sensors

**Figure 5.7:** Expanded View Connection/Disconnection appropriate Gateway

### 5.2.3 Imperative Modeling Solution of the 3rd Use Case

The 3rd use case called *Detection of Point of Failures* was related to introduce capabilities in the system to react to bad performance detection and try to find solutions in its close environment.

To model this use case it is needed before to analyze which events could happen during the execution of the system.

- Alive message: This type of messages will be received when the connected server is available so nothing would be necessary to be done.
- Compensation events: Connection and installation operations of different entities can provoke errors so it is necessary to handle them and compensate the changes in order to not affect in the performance or in the availability of these entities.
- Connection and disconnection errors events: There will be cases in which the (dis)connection between the device and the server could be denied. In this case, it will be notified to the central server which should handle the situation.

In the Figure 5.9 we present a modeling solution for this use case. The default situation in this use case is that a node (NA) is connected with a server instance (SA). So in this workflow, the two initial activities are the start of the execution of NA and the establishment of the connection with the SA instance. Then alive messages from the server instance need to be received. When a time threshold is exceed without not receiving any answer, it starts the disconnection process from this instance and after detecting the most efficient node instance, NA is connected with it and it would be maintained until a new time-threshold is exceeded the process would be started again.

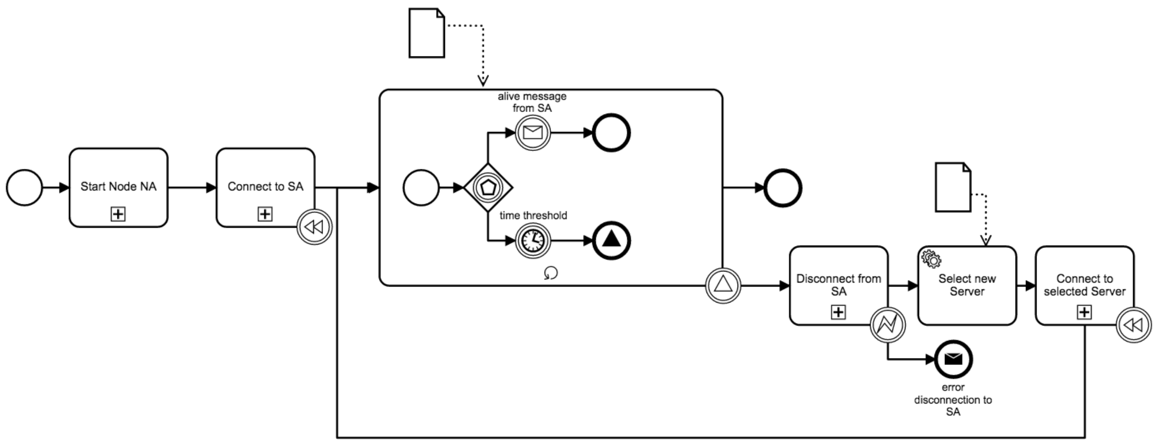


**Figure 5.8:** Imperative General Model Approach of 3rd Use Case

As an alternative to this use case, it may happen that the resources in which a node instance is deployed do not work as expected. In that case, it would be necessary to select a new resource available in the topology or an external one to install the node instance so that the performance improves. In the Figure 5.10 it is represented how this alternative would be modeled. As a difference with the general use case, in this case instead of receiving alive messages from the server which is connected, it would be analyzed the performance by taking into account device context. If the performance is good, there would be no changes in the topology. Nevertheless, when a bad performance is detected, first of all the uninstalling of the node instance is executed and then after finding a resource in which it could be deployed, the configuration and installation would happen. This would be maintained until a new bad performance is detected which trigger again the process.

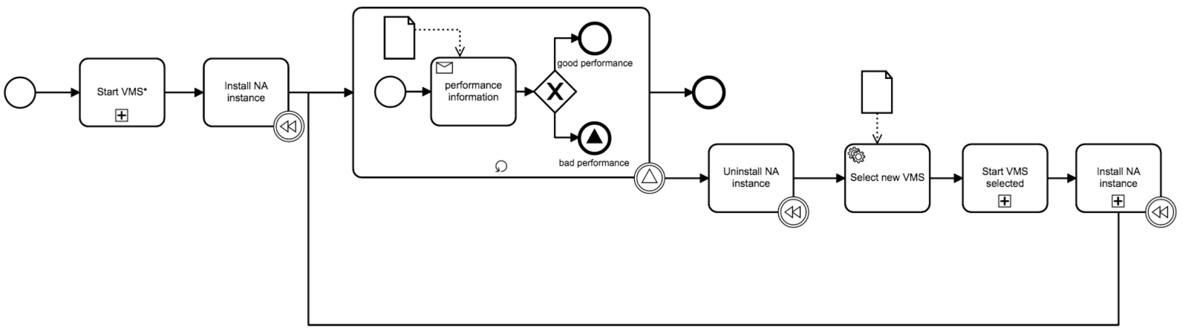


**Figure 5.9:** Imperative Alternative Model Approach of 3rd Use Case

### 5.2.4 Imperative Modeling Solution of the 4th Use Case

The 4th use case called *Video Streaming* is related to take advance of the available resources in the closeness to improve the performance of the service provided. In this case it is focused in the selection of the channel to broadcast multimedia content depending of the resources available and the quality of the connection to them.

The different events associated with this use case are the following:

- Video Server Detection: This message would be received and would trigger the change of sending streaming, not by the video device, instead by the video server which have good connection to Internet.
- Compensation events: Connection and start operations of different entities can provoke errors so it is necessary to handle them and compensate the changes in order to not affect the performance or the availability of these entities.
- Bad connection detection: When it is detected that the quality of streaming by Internet is bad enough it would be disconnected from the server and the streaming would be sent by the traditional way.

In the Figure 5.10 it is represented how this use case would be implemented. First of all the video provider will be started and the traditional streaming will be initiated. This should be maintained until a video server with good stream capabilities is detected and a good WIFI connection is available. If this happens, the video provider is connected with this server and the video streaming is handled by that server. When it is detected that the Internet Connection is decreased, the video provider would disconnect from the video server and the traditional streaming would rebooted.

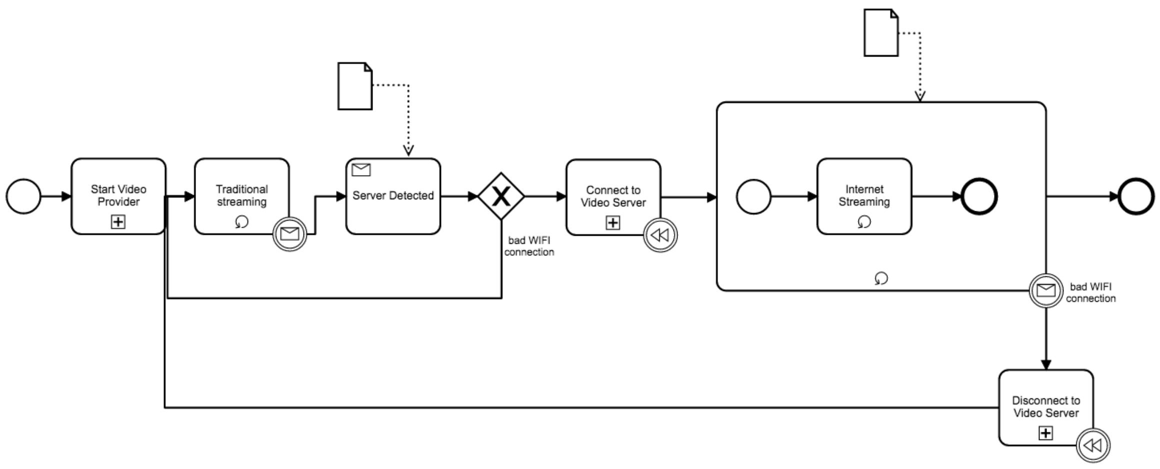


**Figure 5.10:** Imperative Alternative Model Approach of 4th Use Case

### 5.2.5 Imperative Modeling Solution of the 5th Use Case

The 5th use case called *Emergency Situation* is related to react as quickly as possible to situations which need to be handled by emergency servers because of its harmful impact.

The different events associated with this use case are the following:

- Emergency Situation Detection: This message would be received and would trigger the connection to the emergency server in change of this kind of situations.
- Compensation events: Connection and start operations of different entities can provoke errors so it is necessary to handle them and compensate the changes in order to not affect the performance or the availability of these entities.
- Emergency Situation deactivation: When it is detected that the emergency has been resolved, the connection with the emergency server would be disabled, getting to the normal state of the topology.

In the Figure 5.11 it is presented how this use case could be modeled. When the execution of the service is started, a monitoring entity would be instantiated and initiated. When a emergency situation is detected, a quick connection with the appropriate emergency server is established and the monitoring entity would send data recollected by the service until the emergency situation is handled and the alarm deactivated, when ends the connection with the emergency server. This process have priority from other processes that are being executed simultaneously so events associated to emergency situations would be treated before other events.

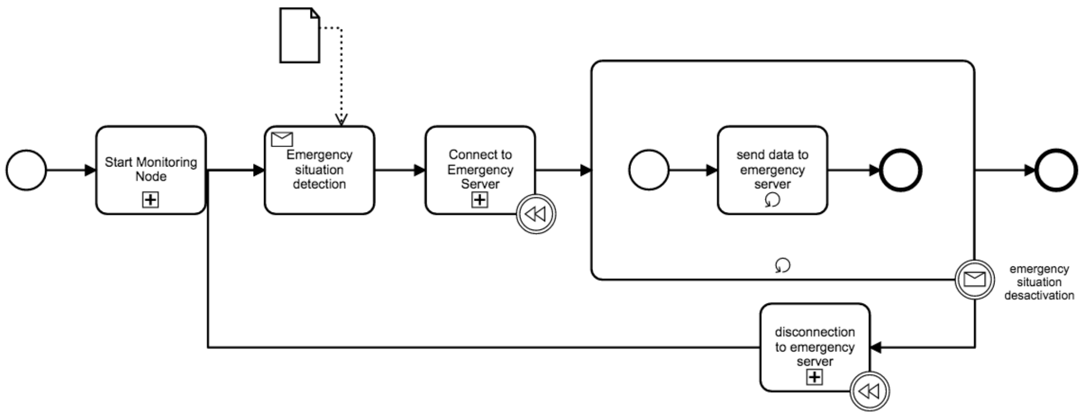

Start Monitoring Node
Emergency situation detection
Connect to Emergency Server
send data to emergency server
emergency situation desactivation
disconnection to emergency server

**Figure 5.11:** Imperative Alternative Model Approach of 5th Use Case

### 5.2.6 Conclusions of Imperative Approach

After analyzing how it could be modeled different use cases of the general scenario described in the previous chapter, we have reached to some conclusions:

- **Good adaptability to customize** :Currently, the amount of services provided to the user is bigger than it used to be. Moreover, an addition value wants to be given so new services are characterized for being customized by each user. In this use cases, it has been shown that every detail can be given of how a service should be provided such as specifying that a service should be provided by a specific server template or which gateway the node should connect and what timing.
- **Easy to Monitor**: As in the imperative approach, all the possible behaviours that a system could have, is easy to detect, such as, possible failures in performance or situations that may need a manual intervention. For example, errors in the installation of nodes in the topology or connection with others could be solved in a easy way by using this approach.
- **Lack of flexibility in run-time**: As it was said before, it is needed to define all the details before the execution of the processes in reference to the performance of the system. This means that it is required to cover details such as all the possible situations that may occur or what the system need to behave in each of those situations. In the use cases where in some situations there was needed to connect to new components, using the Imperative Approach it was necessary to define before a list of nodes to be connected in order to achieve the selection. In the examples

we introduced it would be possible to use an imperative approach, but if the use cases were more complex by introducing more components or more situations possible, which could be unpredictable, during the execution, the complexity of modeling would be a big challenge.

## 5.3 Concepts Declarative Approach

All the modeling topologies which will be presented in this section have a single node in common. As it was introduced in the Section 2.2 the gateway XOR of the BPMN was a clear candidate to be used in the declarative processed topologies. This gateway, modified for TOSCA, would trigger the change of a topology when a new context situation. By that, this node would be in charge of being aware of changes in the context. We explain in the Mapping Section (see Section 5.4) in detail how this would be achieved. This node would execute algorithms evaluating measures of the context to detect if a change has occurred. In that case,it would return a trigger to the worflow to adapt to the context.

Following the structure of the previous section, it would be presented the solutions designed for the use cases.

### 5.3.1 Declarative Modeling Solution of the 1st Use Case

In the Figure 5.12 we present a declarative modeling approach solution for the first use case. In this case, the XOR gateway, depending of the location of the user (Location A or Location B), the server instance would be installed and hosted in the VMs available (VMs A in Location A and VMs B in Location B) and then the connection with the user would be established. One of the rules needed in this declarative model is that the connection would be established after the installation and initiation of the server instance in the VMs selected. By using this approach, it would not be necessary to pre-define the VMs which are going to be selected depending on the location, it could be detected and selected in runtime, as a main difference with the imperative solution presented.

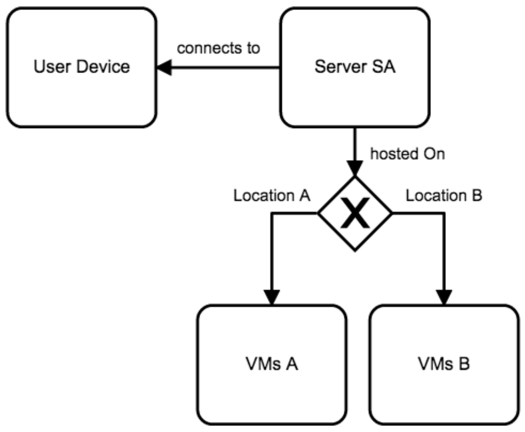


**Figure 5.12:** Declarative General Model Approach of 1st Use Case

When the system is initiated, it will not install any node until it detects the location of the user. When it is detected, the decision point, XOR gateway, would install and start the topology associated to the location which would be composed by the VMs associated to the location, an instance of the server hosted in the VMs selected and required to provide the service as a common node in both topologies, and would represent the connection to the user, which should be established when the previous components are correctly started in the topology. When a change of location is detected, first of all the connection with the user will be disabled, then it would uninstall the instance of the server, stop the performance of the VMs selected and then it would initiate the other topology by starting and installing all the nodes required. In the Figure 5.13 we represent all the possible states of the topology associated to this use case.

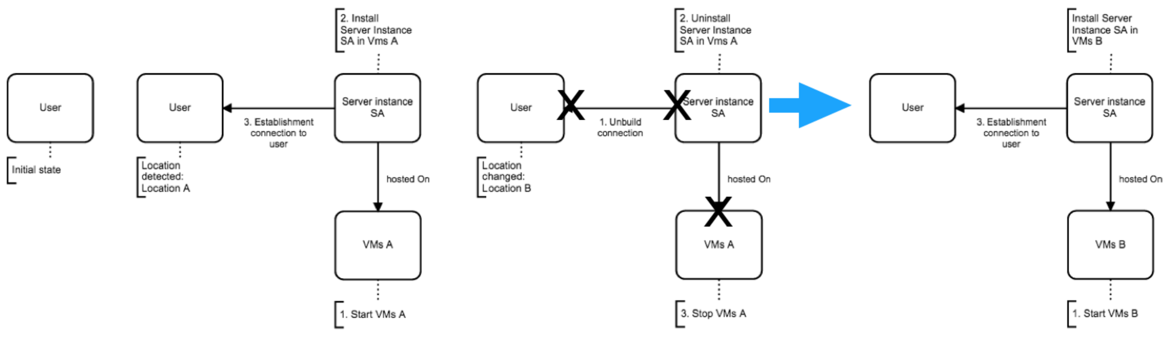

User
Initial state
User
Location detected: Location A
3. Establishment connection to user
2. Install Server Instance SA in Vms A
Server instance SA
hosted On
VMs A
1. Start VMs A
User
Location changed: Location B
1. Unbuild connection
2. Uninstall Server Instance SA in Vms A
Server instance SA
hosted On
VMs A
3. Stop VMs A
User
3. Establishment connection to user
Install Server Instance SA in VMs B
Server instance SA
hosted On
VMs B
1. Start VMs B

**Figure 5.13:** Possible states of 1st case topology during run-time

In the Figure 5.14 we present the declarative modeling solution of the alternative proposed in this use case: a service could be provided by different server templates. In this case, it would be necessary two XOR gateways, which depending of the location, find the VMs which would host the server instance and then install the server instance, appropriate to the location, and finally the connection to the user with this instance would be established. As well as the general use case, it must define some rules associated with the order of the installation of the nodes which are going to be integrated in the topology: the establishment of the communication to the user must have as a precedence the installation and initiation of the Server Instance and the precedence of the Server Instance would be the selection of the VMs in which it would be hosted.

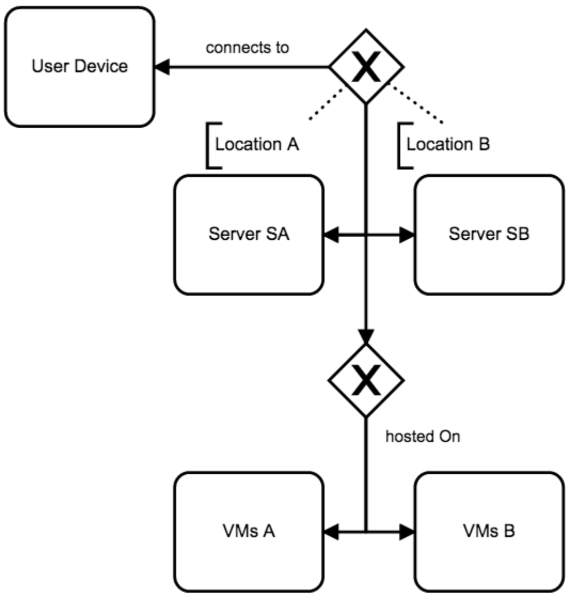


**Figure 5.14:** Declarative Alternative Model Approach of 1th Use Case

As well as the previous use case, it will not be installed until the location of the user is detected. In that moment it will start the VMs associated to that location, the instance of the server available in that location will be installed and the communication with the user and the server instance which is going to provide the service that he is consumer of, will be established. When a change of location happens, the procedure will be the same as the previous case, first of all the communication would be disabled, then the uninstalling of the instance and finally the stop of the VMs of the previous location. When the unbuild is done, then the installation of the new topology could be executed. In the Figure 5.15 these states are represented:

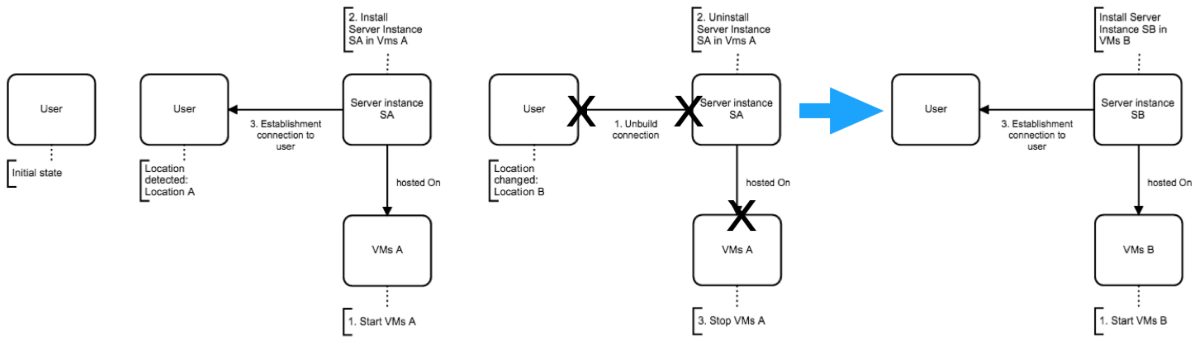

User
Initial state
User
Location detected: Location A
3. Establishment connection to user
2. Install Server Instance SA in Vms A
Server instance SA
hosted On
VMs A
1. Start VMs A
User
Location changed: Location B
1. Unbuild connection
2. Uninstall Server Instance SA in Vms A
Server instance SA
hosted On
VMs A
3. Stop VMs A
User
3. Establishment connection to user
Install Server Instance SB in VMs B
Server instance SB
hosted On
VMs B
1. Start VMs B

**Figure 5.15:** Possible states of 1st case(alternative) topology during run-time

## 5.3.2 Declarative Modeling Solution of the 2nd Use Case

In the Figure 5.16 we present the declarative modeling solution of the second use case. In this use case, the gateway XOR would select the gateway the environment monitoring node should connect to. When the time frame changes, the XOR would execute an algorithm whose result would be the id of the gateway of the sensor of this time frame. This use case could be extended to more time frames or simultaneous connections with gateways of different environment measures.

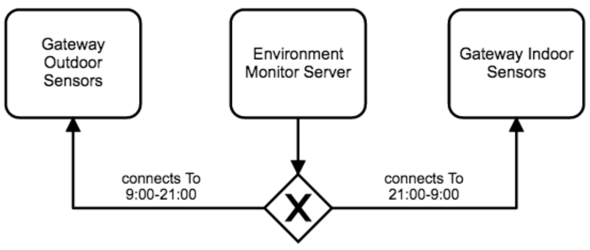


**Figure 5.16:** Declarative General Model Approach of 2nd Use Case

In this case, as a difference with the previous solutions, before detecting the context it would install an Environment Monitor Server Instance. Then, it would check what time frame is and the connection with the appropriate gateway would be done. When a change of time frame is detected, then first of all it would disable the communication with the previous gateway and then it would establish the new communication. In the Figure 5.17 we present all the states of the topology that can occur during the execution.

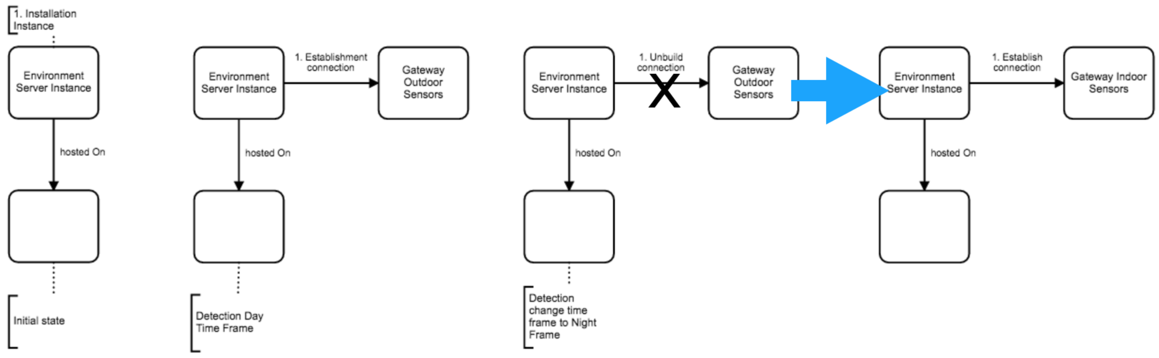


**Figure 5.17:** Possible states of 2nd case topology during run-time

### 5.3.3 Declarative Modeling Solution of the 3rd Use Case

In the Figure 5.18 we present the declarative modeling solution of the third use case. In this use case, a default topology is introduced when the run-time is started. The XOR gateway, when it is detected that the server instance(SA) to which the node NA is connected is not working as it was expected, executes an algorithm in which selects the id of the new server instance to be connected. With this approach, it will not be necessary to pre-define which server the node to connect to, it will be only necessary to specify which capabilities are required.

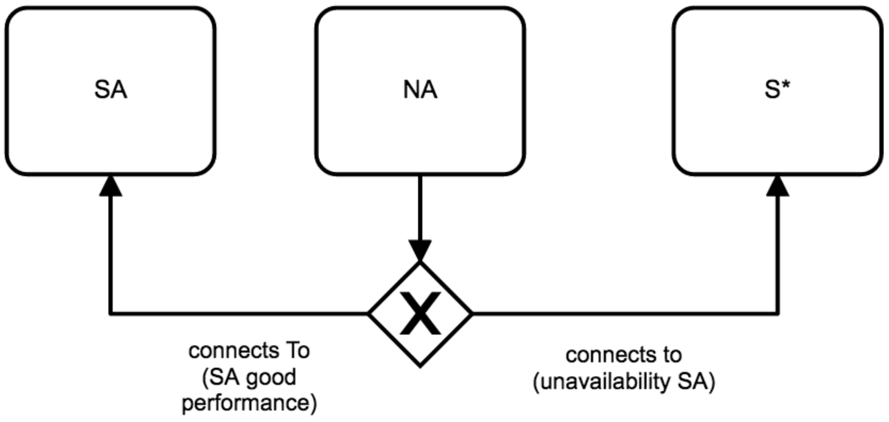

SA
NA
S*
X
connects To (SA good performance)
connects to (unavailability SA)

**Figure 5.18:** Declarative General Model Approach of 3rd Use Case

In this case, as the previous one, an initial topology will be installed when the system is initiated. This topology only would be changed if a bad performance of the server instance is detected. In that case, first of all, it would disable the communication with it and then after deciding which server instance, it would replace a new communication would be established. In the Figure 5.19 it is presented all the possible states that the topology of the solution could have during the execution.

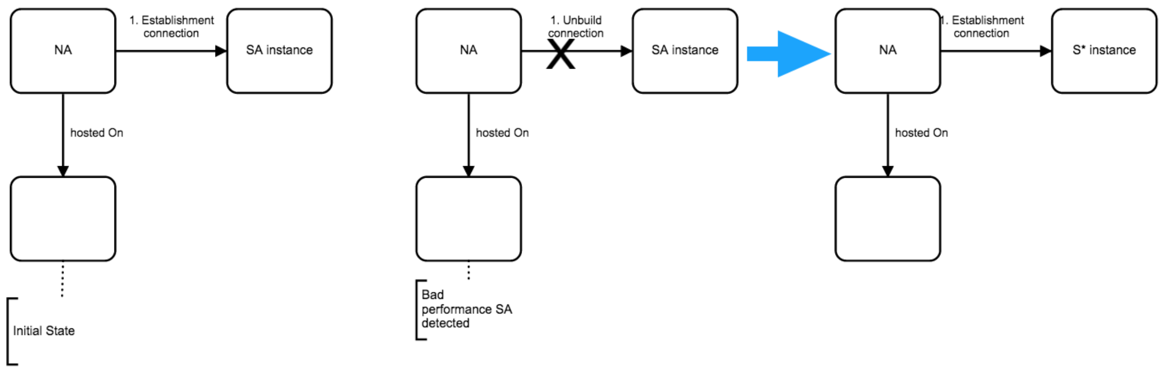


**Figure 5.19:** Possible states of 3rd case topology during run-time

As an alternative it was analyzed that instead of detecting bad performance of communications with external nodes, detecting the performance of the resources in which the node instance are installed. In the Figure 5.20 we present the declarative modeling solution of this alternative. In this case, the XOR gateway would handle the installation of the node instance in the VMs which has better performance.

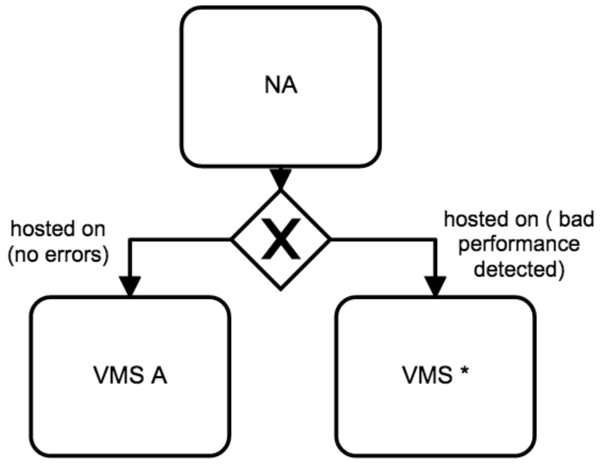


**Figure 5.20:** Declarative Alternative Model Approach of 3rd Use Case

As well as the general case, a default topology would be installed: a node instance NA is installed in the VMs A. When it detects that the performance of the VMs A is not good enough, first of all the node instance is uninstalled from the VMs A, this resource stops its execution and then after starting the new resource in which it is going to be deployed the instancing (VMs*) the NA instance would be again installed but in this new resource selected. In the Figure 5.21 we present all possible states of the topology that they could occur during the run-time.

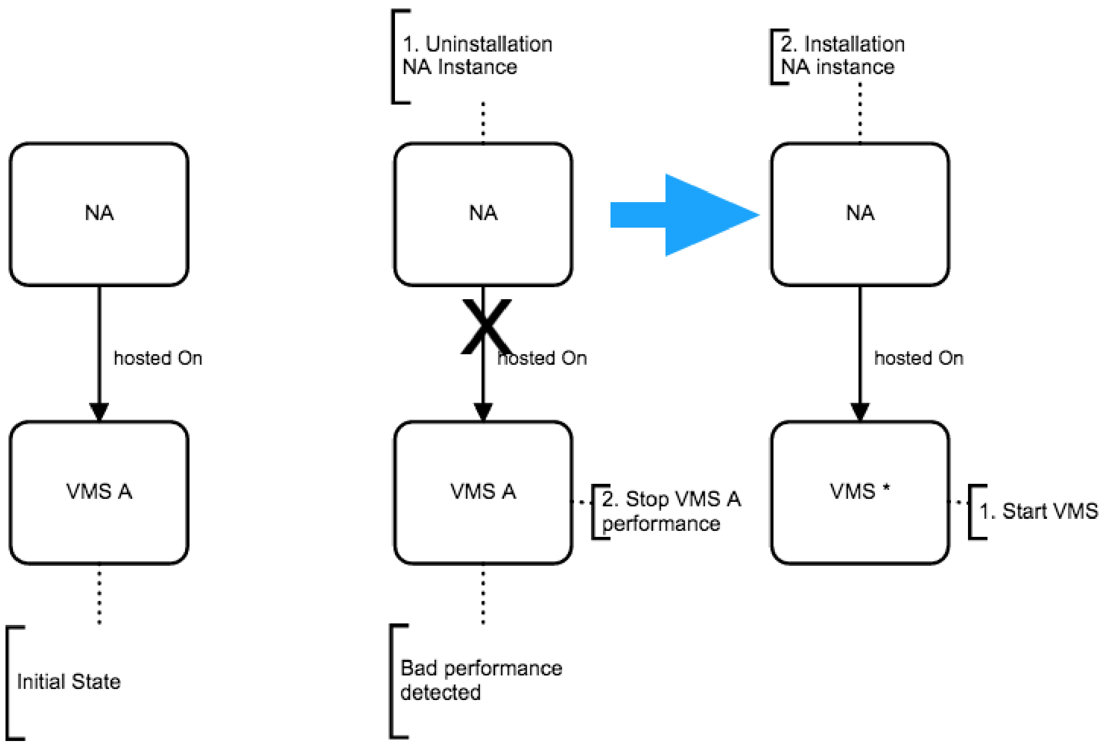


**Figure 5.21:** Possible states of 3rd case(alternative) topology during run-time

### 5.3.4 Declarative Modeling Solution of the 4th and 5th Use Cases

In the Figures 5.22 and 5.23 we present the declarative modeling solutions for the forth and fifth use case. They have in common that the XOR depending of the context situation in which the system is would decide if a new relationship between a node of the topology with an external node is established (video server or emergency server depending of the use case).

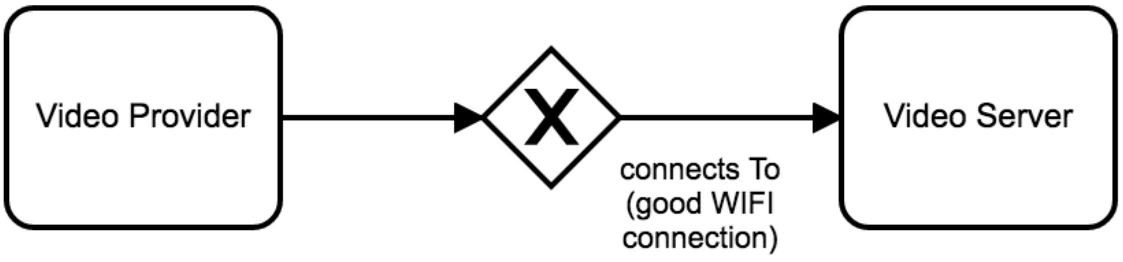


**Figure 5.22:** Declarative General Model Approach of 4th Use Case

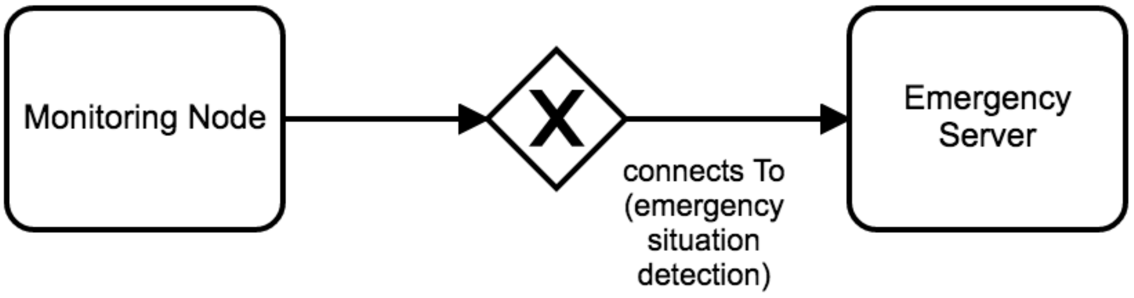


**Figure 5.23:** Declarative General Model Approach of 5th Use Case

In these cases, a default topology will be installed when the system is started. In the 4th case it should be configured and start the execution of the Video Provider and in the 5th case the Monitoring Node. When the condition is detected ( good WIFI connection / emergency situation detection) the connection with the respectively server instance will be established. When the condition, which has triggered the change in the topology is not fulfilled any more, the connection is disabled so the topology returns to the initial state. In the Figure 5.24 we represent all the possible states of the topology that could occur during the execution.

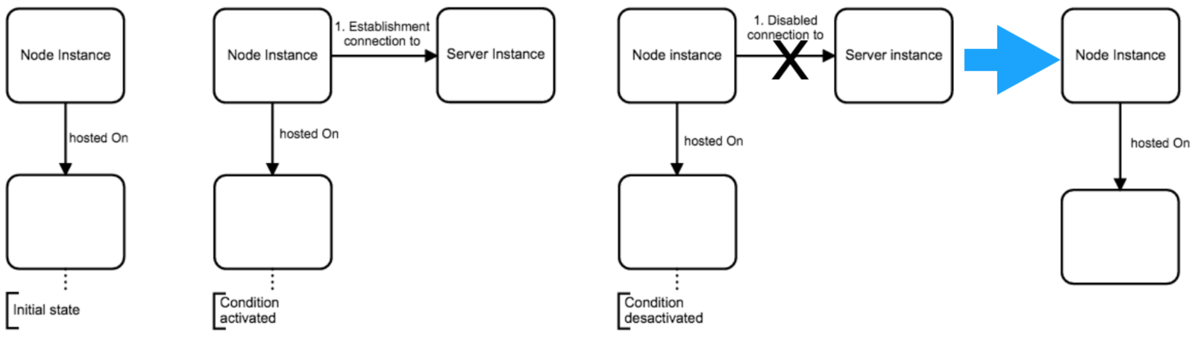


**Figure 5.24:** Possible states of 4th and 5th topology during run-time

### 5.3.5 Conclusions of Declarative Approach

After analyzing how it could be model the different use cases proposed in a declarative approach, we have reached to these conclusions:

- **Easy to model unforeseen situations**: Context-aware systems are characterized by being adaptive to multiple context situations. In the imperative approach it is required to pre-define the possible situations that can occur. By contrast, with the declarative approach this is not required so the solution modeled could cover all the states of the system during the execution. Moreover, the design of the solution is simpler than the imperative one because all the details of how the system should react when a change of context is detected are not required to be integrated in the solution. As it was said before, the use cases proposed could cover more locations, time-frames or more node instances to connect and the complexity of the solution would not vary.
- **Easy to model Complex systems**: Context-aware systems are characterized by being composed by a big amount of different entities which are interconnected by temporal or situation-aware communications. It has been seen in some use cases that depending of the context such as time-frames, emergency situations or network conditions the communication with different server instances are established. By using this approach, this could be easy modeled with the use of qualified decision points which are which are in charged of depending of the situation trigger changes in the topology.
- **Deficiencies in monitoring**: As it is not defined the performance of the system by using this approach, is hard to monitor to detect how efficient the system is during execution and possible errors are more difficult to detect than in the imperative solutions.
- **Lack of customization services**: Declarative programming is focused on standard and portable services, but as it is not easy specify the details of behaviour of the system, it cannot easily specified customization in performance. Nevertheless, the use cases described were able to be modeled by the declarative approach because changes in context trigger changes in communications and incorporation of new nodes in the topology are standard processes.

## 5.4 Mapping

In order to generate the events which trigger the changes into the topologies of the different solutions, it has been defined an architecture which allows the adaptation to situation changes of the different services which integrate a context-aware system. This architecture is based on the solution proposed by L. González et al. to construct context aware systems [GO14]. In the Figure 5.25 we present the high-level specification of the architecture.

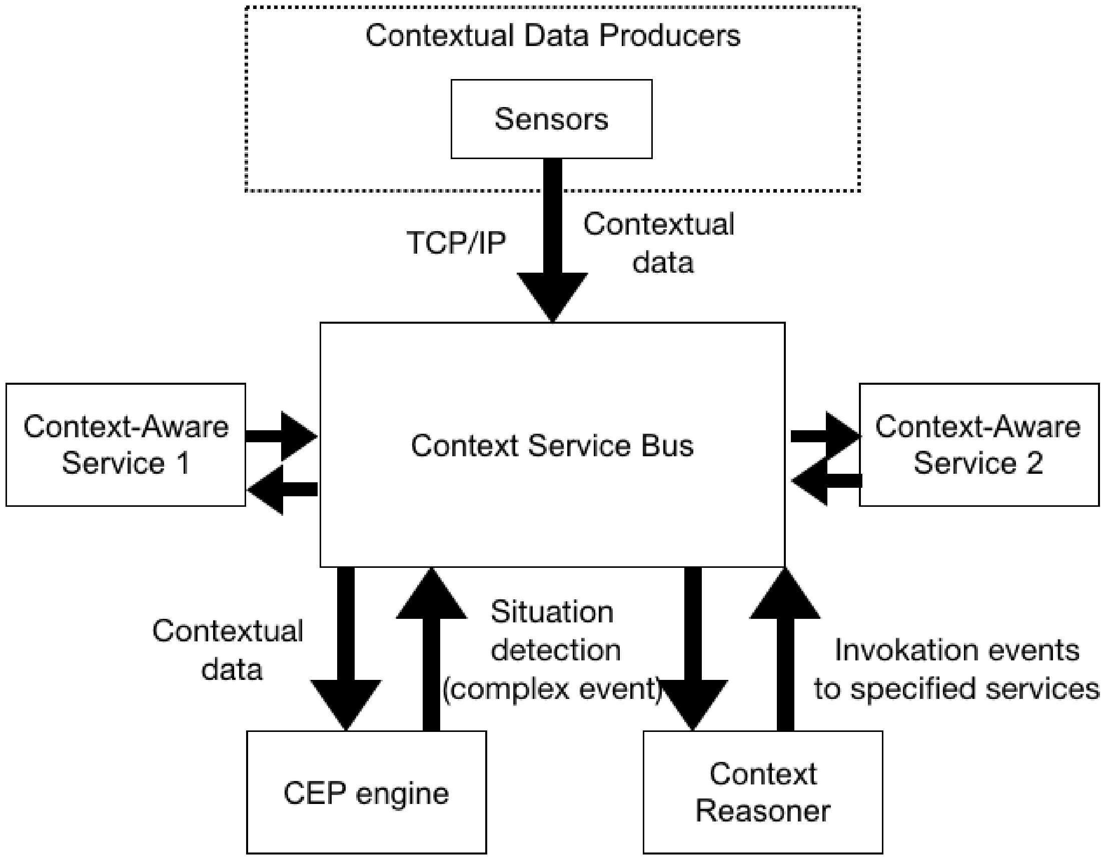


**Figure 5.25:** General Architecture

As it can be seen in the figure, it is integrated by 4 principal components:

- Contextual Data Providers: This component represent the external sources which provide during the run-time contextual information to the system.

- Complex Event Processing (CEP) engine: CEP techniques are used to analyze the contextual information introduced by the contextual data providers and detect complex situations that might affect the services of the system.
- Context Reasoner: This component knows what situation is relevant to the services provided in the system (use cases) and trigger the events required to these services. This component was called Monitoring Entity in the introduction of the section 5.2.
- Context Service Bus: This component which is in charge of exchanging all the messages from the different components of the architecture and also delivering the appropriate events to the different services in the system.

To model the integration of the context into Service Templates in TOSCA, first of all it is necessary to analyze the possible options to process the data taken by the different sources a context-aware system has:

- Data shipping: This approach is based on retrieving data from the data location and processing it where the computational resources resides. This approach is characterized by high communication costs because of the size of raw data that should be sent, but the level of customization in the processing and the independence of the location of the data are also factors to be taken into account where selecting this approach,
- Function Shipping: This approach is based on executing functions (queries) on the data location and only sending the result of the operation to the client side ( context-aware system). By that, a reduction of the computation in the client side is fulfilled but it is necessary to locate the computation resources as close as possible to the data storage.

Because of the benefits of each approach and to give to our solution flexibility both approaches have been considered. Saatkamp introduced 8 modeling concepts which enables these approaches in TOSCA [Saa16]. After analyzing these solutions, we have selected the approaches that could implement both data and function shipping and integrated them into two final solutions which differs mainly of the unique or multiple data sources that could have the system.

- **Data connector between Processing Logic and unique Data Resource**: This concept focus on the relationship that has to be modeled in order to communicate the Processing Logic Entity with where is located the context information. In the Figure 5.26 it is represented the basis of the solution. It can be seen that it is needed to define in TOSCA three components, the Processing Logic Entity which identifies the entity which is in charge of processing the data, the Data Connector Relationship which represents an 1:1 relationship between a Processing Logic and a Data Resource and finally a Context Data Resource which identifies the entity where it is locating the data. Depending of the approach selected, it would include the Deployment Artifact with the required functions to execute in the data resource (Function Shipping) or the location address where is placed the data resource (Data Shipping). Moreover, it may integrate in the model additional capacities such as transformation operations of data. In order to do that, it would be necessary to link this operations to the Data Connector Entity and introduce properties about the structure and format of the data storage.

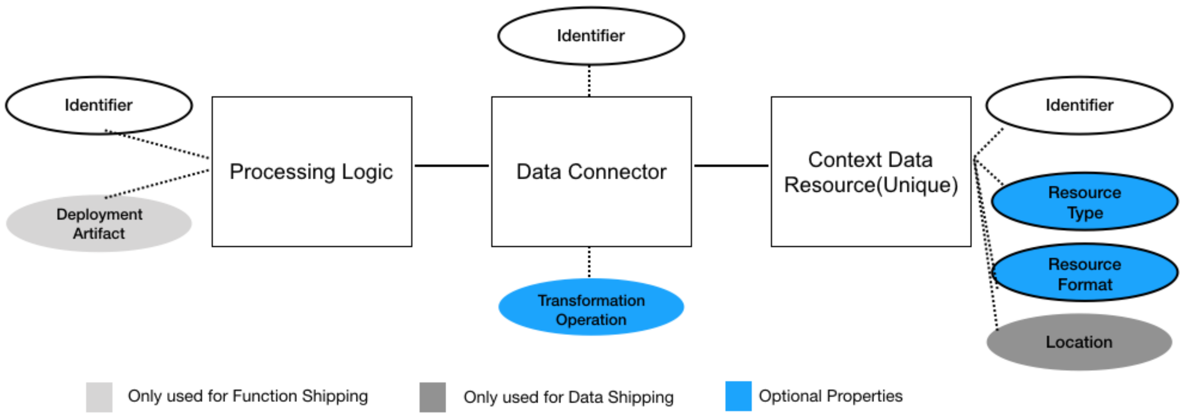


**Figure 5.26:** 1 Concept: Data connector between Processing Logic and unique Data Resource

Although different TOSCA realizations were proposed by Saatkamp to this concept, we have selected the realization which integrates data queries. As this concept only considers an unique data resource in which all the context measures would be integrated, it is necessary that each service only has access to the relevant information. In the Figure 5.27 we present how modeling the topology template of the first use case described in the Chapter 4 is possible, whose relevant context information was spatial context. First of all, it is needed to define two node types: Processing Logic and Context Resource. The Processing Logic Node Type provides the operation *checkchanges* which takes a context measure as a input via the Application Interface *Context Change Detection Interface*. In the Listing 5.1 the definition of this Node Type is represented. The operation's input would be a location variable which is an array composed by two boolean related to the two possible locations (Location A and Location B).

**Listing 5.1** Processing Logic Node Type Definition

```
< Node Type name="ProcessingLogic">
...
   <opentosca:ApplicationInterfaces
       xmlns:opentosca="http//www.uni-stuttgart.de/opentosca">
      <Interface name = "Context_Changes_Detection_Interface">
         <Operation name="checkchanges">
            <documentation> detects changes of location of users registered in the
                service </documentation>
            <InputParameters>
               <InputParameter name="location">
                  <xs:complexType>
                     <xs:attribute name="LocationA" type="xs:boolean"/>
                     <xs:attribute name="LocationB" type="xs:boolean"/>
                  </xs:complexType>
               </InputParameter>
            </InputParameters>
         </Operation>
      </Interface>
   </opentosca:ApplicationInterfaces>
</NodeType>
```

The Context Resource Node Type defines three properties: Resource Type (R.Type in the Figure) describes what the structure of the data storage is, Resource Format (R.Format in the Figure) describes what format is available for the data and the Location references where the data storage is. In the Listing 5.2 the definition of this Node Type is represented.

**Listing 5.2** Context Resource Node Type Definition

```
<NodeTemplate id="ContextResourceFile" name = "Context Resource File" type="DataResource"
...
   <Properties>
      <DataResourceProperties>
         <Properties>
            <xs:element name="R.Type" type="xs:String">/>
            <xs:element name="R.Format" type="xs:String>/>
         </Properties>
         <Location ref="address" minOccurs="0">/>
      </DataResourceProperties>
   </Properties>
</NodeTemplate>
```

Moreover, it is necessary to define the *DataConnector* Relationship Type which has 4 properties: *Source Interface, Source Operation* and SourceInputPar which are the identifiers of what operation of what interface is going to be used and what input data is required to such operation, and *DataQuery* in which can be specified a query of what data is required from the Context Resource. In the Listing 5.3 the definition of this Relationship Type is represented.

**Listing 5.3** Data Connector Type Definition

```
<RelationshipTemplate id="DataConnectorFile" name="Data Connector File"
    type="DataConnector">
...
   <Properties>
      <DataConnectorProperties>
         <DataConnectorProperty name="SourceInterface" type="xs:String">/>
         <DataConnectorProperty name="SourceOperation" type="xs:String">/>
         <DataConnectorProperty name="SourceInputPar" type="xs:String">/>
         <DataConnectorProperty name="DataQuery" type="xs:String">/>
      </DataConnectorProperties>
   </Properties>
</RelationshipTemplate>
```

After defining the typing, it is necessary to define the Topology Template. This would be integrated by the following components.

- **Location Change Detection**: This template is a Processing Logic Node. If the approach used is Function Shipping, it is needed to include a Deployment Artifact to deploy and run the processing logic.
- **Context Resource File**: This template is a Context Resource Node. In this use case, the context data is stored in a SQL table. This is defined in the properties R.Type and R.Format. Moreover, if the approach used is Data Shipping, it is needed to include the location of the data resource file.
- **Data Connector Table**: This template is a DataConnector Relationship. This relationship establish the connection between the Processing Logic and Context Resource nodes. Moreover, it defines that it is going to be used in the operation *checkchanges* of the interface *LocationInterface* and that the input required by this operation would be the user location.

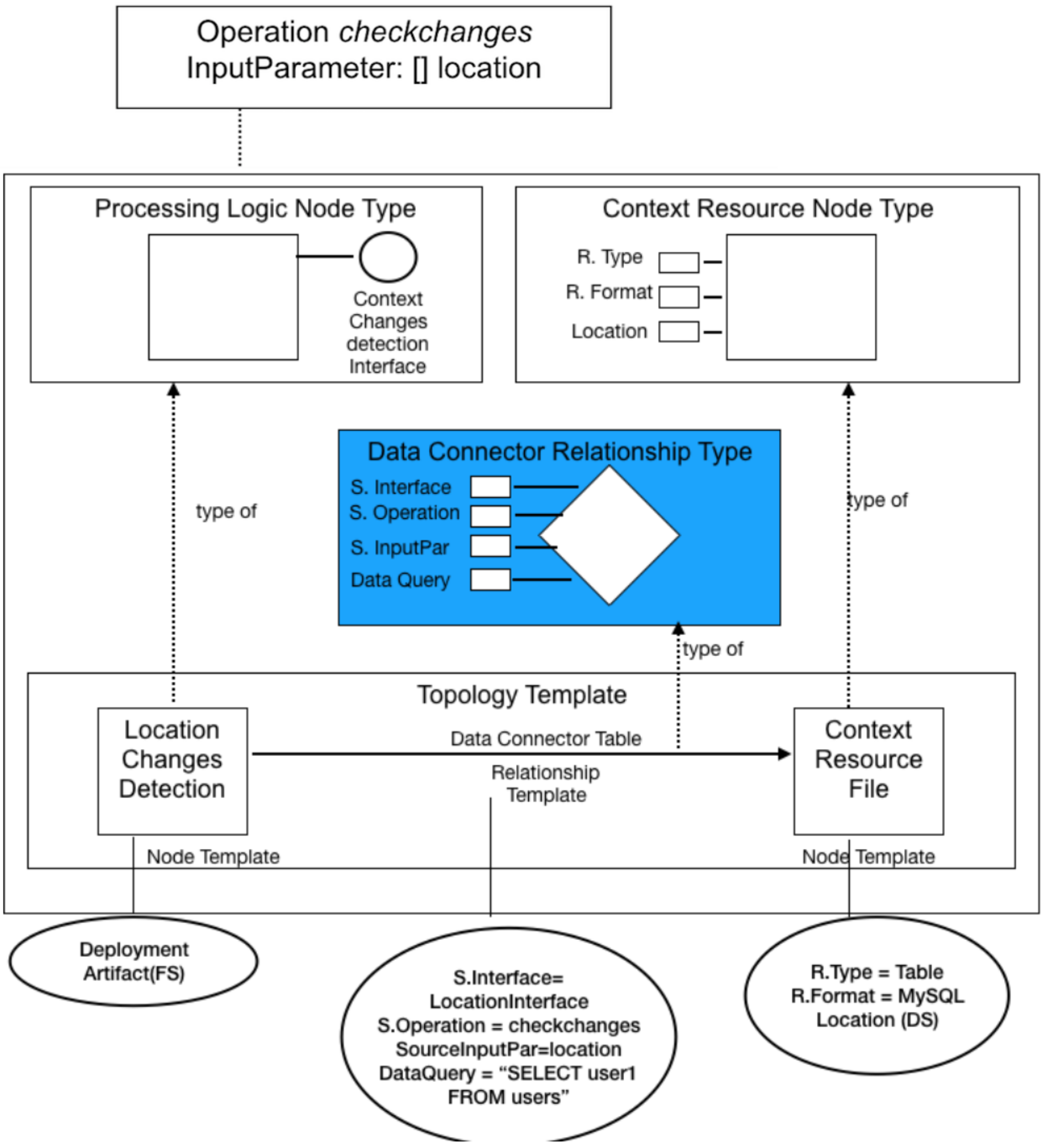


**Figure 5.27:** TOSCA realization with explicitly modeled data query

- **Data Connector with Operations Applied to Multiple Data Resources**: By contrast with the first concept, in this approach it can use different data resources which can be modeled as data collection if they belong to the same collection and moreover it can have operations to be applied in these resources and to connect them to a processing logic. In the Figure 5.28 we present the basis of the solution. As well as in the first approach, the Processing Logic Node would have as properties an identifier and if it uses Function Shipping the Deployment Artifact Required. The Data Connector Entity connects the Processing Logic Node with the different Data Resources available, for that reason, it is assigned a Join Operation attribute which allows to join data from the different sources together with the aim of providing a single input for the processing logic node. The Context Data Collection may comprise several data resources and it is characterized by four attributes: identifier and location, as well as in the first concept, and Collection Type and Resource Type which defines the type of collection it is ,such as directories or databases, and the type of data resources hold in the collection respectively.

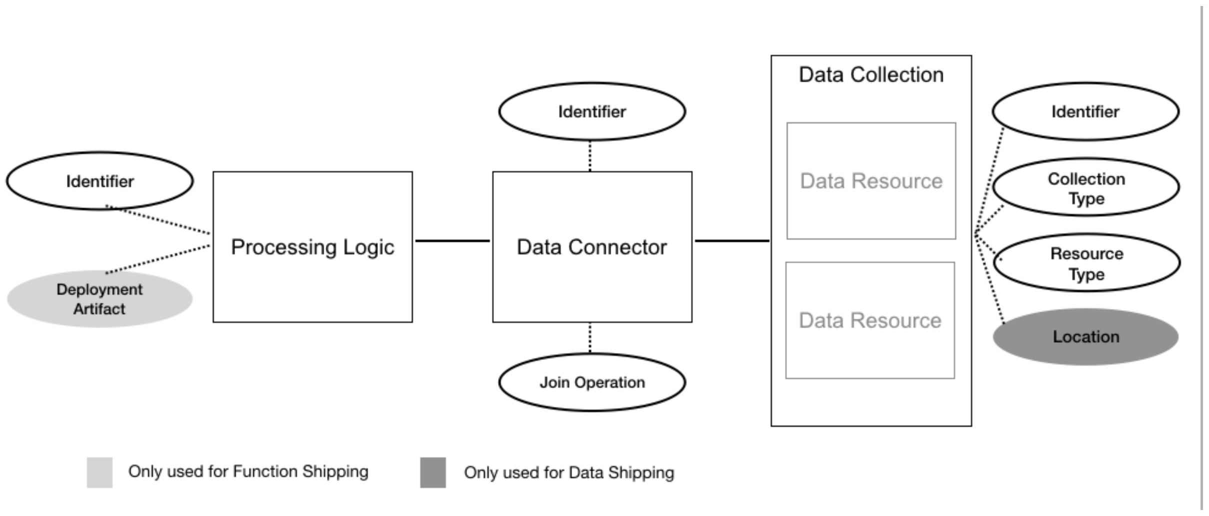


**Figure 5.28:** Data Connector with Operations Applied to Multiple Data Resources

Although there were proposed alternatives to model this concept, we have selected the solution based on assignment of the Join Operation via a Join Interface. In the Figure **??** it is represented how it would be modeled this solution. As well as the first solution proposed, the topology would be integrated by two node templates and a relationship template. By that first, of all it is necessary to define the types associated to these templates. The definition of the Processing Logic Node would be equal to the proposed in the first solution and described in the Listing 5.1. A new Node Type is defined in this solution: Data Collection. As commented before, the definition of the type would require to include three attributes: collection type, resource type and location if Data shipping is used. Moreover, it would include the resources which are included in the collection. Listing 5.4 includes the definition of this node type.

**Listing 5.4** Context Collection Node Type Definition

```
<NodeTemplate id="ContextResourceFile" name = "Context Resource File" type="DataResource"
...
   <Properties>
      <DataResourceProperties>
         <Properties>
            <xs:element name="Collection Type" type="xs:String"/>
            <xs:element name="R.Format" type="xs:String>/>
         </Properties>
         <Location ref="address" minOccurs="0"/>
      </DataResourceProperties>
   </Properties>
   <ContextResources>
      <ContextResource id="locationresource">
      ....
      </ContextResource>
      <ContextResource id="temporalresource">
      ....
      </ContextResource>
   ....
   </ContextResources>
</NodeTemplate>
```

Finally it is necessary to define a Data Connector Relationship Template. The main difference to the presented first concept is that it includes an interface so-called *Data Join* which introduces the operation *JoinData* which aim is to join data from different sources and have as an input an array of Context Ids that want to be processed in the Processing Logic Node.

After the typing definition, the topology template would be integrated as a template for each node and relationship type. As it can be seen in the Figure, the processing operation that it is going to be used is *checkchanges*, as well as the first solution and moreover, the data collection used would be a directory and the data resources integrated would be files hosted in this directory.

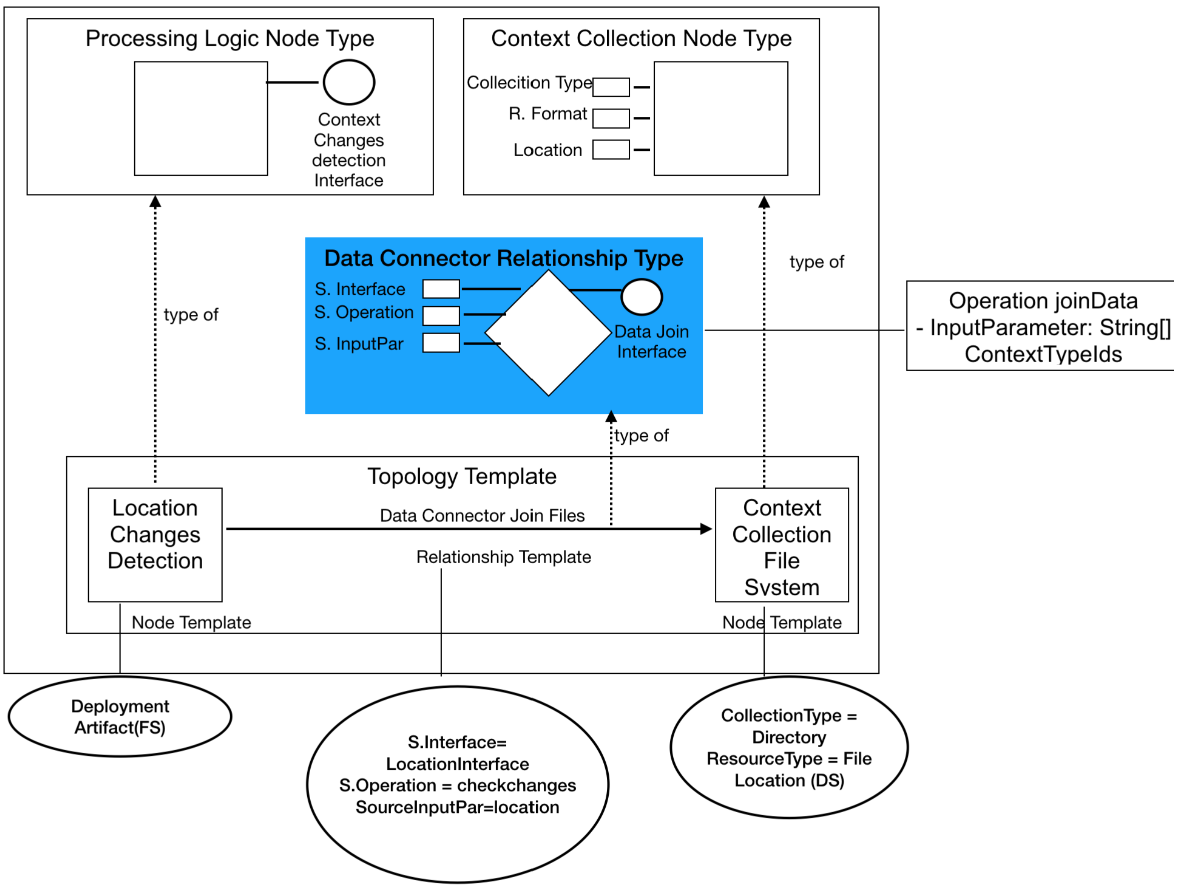

Processing Logic Node Type
Context Changes detection Interface
Context Collection Node Type
Collecition Type
R. Format
Location
Data Connector Relationship Type
S. Interface
S. Operation
S. InputPar
Data Join Interface
Operation joinData
- InputParameter: String[] ContextTypeIds
type of
type of
type of
Topology Template
Location Changes Detection
Data Connector Join Files
Relationship Template
Context Collection File Svstem
Node Template
Node Template
Deployment Artifact(FS)
S.Interface= LocationInterface
S.Operation = checkchanges
SourceInputPar=location
CollectionType = Directory
ResourceType = File
Location (DS)

**Figure 5.29:** TOSCA realization with join operation assignment via join interface

After analyzing modeling solutions which enable the integration of the context into topologies and an architecture solution which adapts the systems deployed, to be aware of changes in the context and to adapt to them, we propose a solution which integrates both and which can be adapted to the different use cases proposed in Chapter 4. In the Figure 5.30 we present the architecture proposed to the first use case, which was related to changes of location. All the components of the architecture would be communicated by the Context Service Bus. First of all, the location sensors would transmit the measures taken into a context data resource, which could be a directory or a single file depending on the modeling solution used. The CEP engine, which is the Processing Logic Node, would execute the operation checkchanges of the interface *LocationInterface* which would detect if the situation has changed. In that case, the Context Reasoner, which is the Monitoring Node in the topology would analyze the new situation and send the appropriate event to the implicated nodes. In this use case, the XOR node would require the notification of change of location to install an instance of the server SA in the VMs associated with the new location (VMs A for Location A, VMs B for Location B).

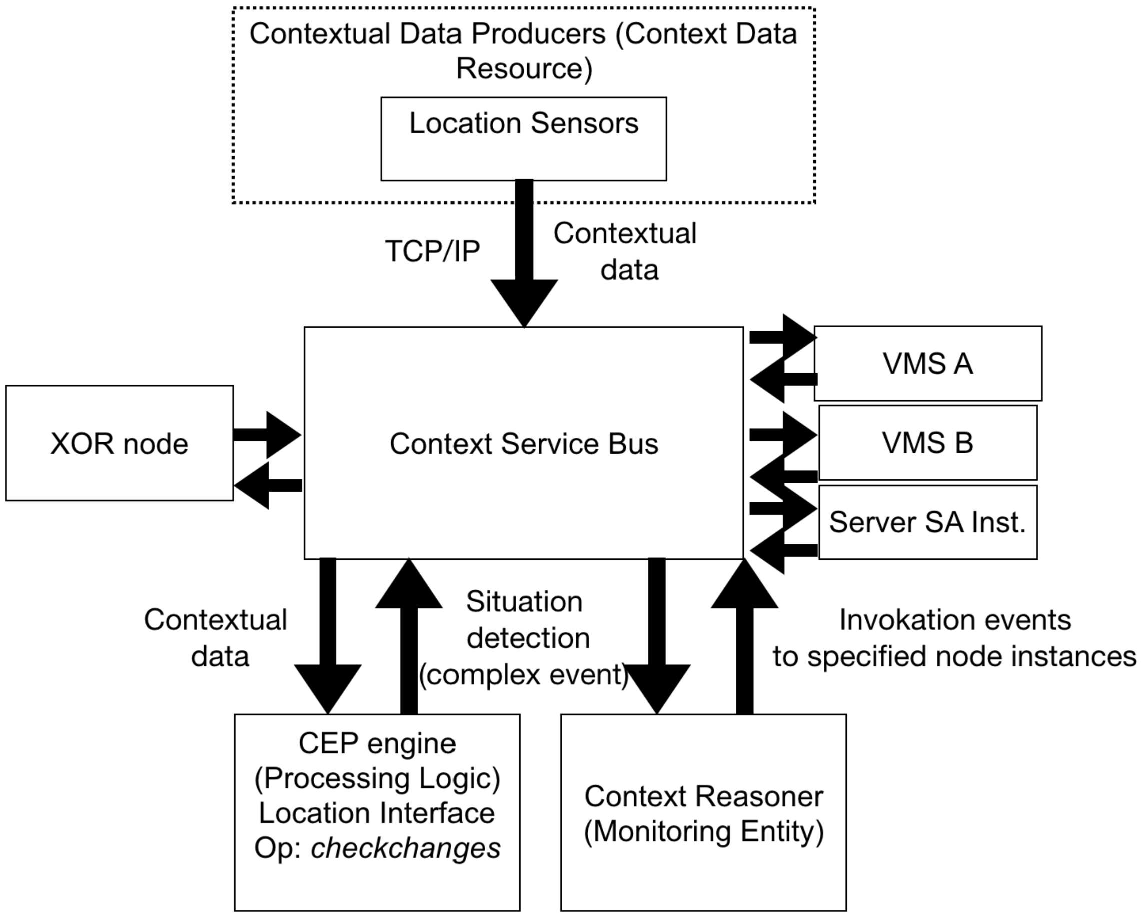

Contextual Data Producers (Context Data Resource)
Location Sensors
TCP/IP
Contextual data
VMS A
VMS B
Server SA Inst.
XOR node
Context Service Bus
Contextual data
Situation detection (complex event)
Invokation events to specified node instances
CEP engine (Processing Logic) Location Interface Op: *checkchanges*
Context Reasoner (Monitoring Entity)

**Figure 5.30:** Integrated architecture for Location Aware System

# 6 Conclusions and future works

The goal of this master thesis is to define a modeling solution for context aware systems using TOSCA. Thereby, modeling context, analyzing different use cases hold in a general scenario which are adaptive to the context, evaluating the requirements needed to model them, designing modeling solutions for them by the use of the modeling languages approaches and mapping this solutions into a general architecture into TOSCA are the main aspects. To model the context we have selected a mark-up model. To model the use cases, the context is integrated by 4 context categories: spatial, temporal, device and network context.In total 5 use cases which use relevant context data for different approaches are presented. 2 modeling solutions of each use case has been proposed by using imperative and declarative approaches and it has been evaluated the suitability of both approaches. On the one hand, by using an imperative approach the customization of the service is easily achieved. On the other hand, by using the declarative approach the flexibility of performance in many situations that may be not able to predict in the design phase is introduced.

Context-aware systems are characterized by reacting to situation changes. In order to trigger events in the different service topologies, we have proposed an architecture solution composed mainly by 4 principal components: a context service bus which is in charge of exchanging information between the different components, a CEP engine which detects situation changes and a Context Reasoner which after being notified of a situation changes, it send events to the affected services templates or entities of the topology. To map the modeling solutions to the use cases and adapt the situation-aware architecture proposed into TOSCA, first of all it is necessary to process the context data taken by the sources of the system. Two approaches have been taken into account for processing the data: data shipping which is based on retrieving data from the data location and processing it where the computational resources resides or function shipping which is based on executing functions on the data location and only sending the result of the operation.

Two final modeling solutions which can be provided in TOSCA have been proposed incorporating in both the chance to choose between the two data processing approaches in order to increase the area in which these solutions can be used. The basis of the two modeling solutions proposed is to define the communication between the Processing

Logic Entity which is going to check the changes of the context by using an operation of an interface defined in the entity with the Context Data Resource Entity which identifies where we located the context information. A third entity is necessary to define in TOSCA: Data Connector which is a Relationship Template and represents the communication. The difference between the two modeling solutions proposed is how the data is storaged. The first modeling solution a unique data source is considered, by contrast the second modeling solution allows the use of multiple data resources. Finally to check its suitability to the use cases proposed, it has adapted the architecture to the location-aware use case.

Further work is required in terms of the development of a runtime environment, which is able to process the presented modeling solution. In order to do that, it can be used in *OpenTOSCA*, a runtime environment able to model application topologies and automatically provide and manage them [BBH+13]. This runtime could be extended with the solution proposed in order to provide the performance required to model context aware applications.

Another field of work is the introduction in the solution, a combination of imperative and declarative approaches in order to include the ability to the developers of context aware systems that can be customized after its generation. As it was introduced in the section 3.3, Uwe Breitenbücher et. al [BBK+14] has proposed a solution to combine both approaches that can be introduced in our solution.

# Bibliography

[AG99] D. K. Anind, A. D. Gregory. "Towards a Better Understanding of Context-Awareness." In: *HUC '99 Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing* (1999), pp. 304–307 (cit. on p. 21).

[All16] T. Allweyer. *BPMN 2.0: Introduction to the Standard for Business Porcess Modeling*. 2016. ISBN: 9783741219788 (cit. on p. 18).

[Bau03] J. Bauer. "Diplomarbeit: Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic." In: (2003) (cit. on p. 28).

[BBH+13] T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, S. Wagner. "OpenTOSCA – A Runtime for TOSCA-based Cloud Applications." In: *11$^{th}$ International Conference on Service-Oriented Computing*. LNCS. Springer, 2013 (cit. on p. 88).

[BBK+14] U. Breitenbücher, T. Binz, K. Képes, O. Kopp, F. Leymann, J. Wettinger. "Combining Declarative and Imperative Cloud Application Provisioning based on TOSCA." In: *Proceedings of the IEEE International Conference on Cloud Engineering (IEEE IC2E 2014)*. IEEE Computer Society, Mar. 2014, pp. 87–96. DOI: DOI10.1109/IC2E.2014.56 (cit. on pp. 44, 45, 88).

[BBK+15a] U. Breitenbücher, T. Binz, O. Kopp, K. Képes, F. Leymann, J. Wettinger. "Hybrid TOSCA Provisioning Plans: Integrating Declarative and Imperative Cloud Application Provisioning Technologies." In: (2015) (cit. on p. 40).

[BBK+15b] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, J. Wettinger. "A Modelling Concept to Integrate Declarative and Imperative Cloud Application Provisioning Technologies." In: *Proceedings of the 5th International Conference on Cloud Computing and Services Science (CLOSER)*. SciTePress, 2015, pp. 487–496 (cit. on p. 44).

[BBKL14] T. Binz, U. Breitenbücher, O. Kopp, F. Leymann. "TOSCA: Portable Automated Deployment and Management of Cloud Applications." In: *Advanced Web Service* (2014), pp. 527–549 (cit. on pp. 35, 37).

[BCB+10] Bettini, Claudio, Brdiczka, Oliver, Henricksen, Karen, Indulska, Jadwiga, D. Nicklas, Ranganathan, Anand, Riboni, Daniele. “A survey of context modelling and reasoning techniques.” In: *Pervasive and Mobile Computing 6.2* (2010), pp. 161–180 (cit. on pp. 22, 25, 27, 28, 31, 34).

[BCQ+07] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, L. Tanca. “A Data-oriented Survey of Context Models.” In: *ACM SIGMOD Record* 36 (2007), pp. 19–26. DOI: 10.1145/1361348.1361353 (cit. on pp. 28, 33).

[BDPP16] A. Botta, W. de Donato, V. Persico, A. Pescapé. “Integration of Cloud Computing and Internet of Things: a Survey.” In: *Future Generation Computer Systems* 56 (2016), pp. 684–700 (cit. on p. 17).

[BDR07] M. Baldauf, S. Dustdar, F. Rosenberg. “A survey on context-aware systems.” In: *Int. J Ad Hoc and Ubiquitous Computing* 2 (2007) (cit. on p. 34).

[BE04] J. J. Bisgaard, D. A. East. *How is Context and Context-awareness Defined and Applied? A Survey of Context-awareness*. 2004 (cit. on p. 21).

[BHH04] S. Buchholz, T. Hamann, G. Hübsch. “Comprehensive Structured Context Profiles (CSCP): Design and Experiences.” In: *Pervasive Computing and Communications Workshops* (2004). DOI: 10.1109/PERCOMW.2004.1276903 (cit. on p. 28).

[CFJ03a] H. Chen, T. Finin, A. Joshi. “An Ontology for Context- Aware Pervasive Computing Environments.” In: *The Knowledge Engineering Review* 18 (2003), pp. 197–207. DOI: 10.1017/S0269888904000025 (cit. on pp. 27, 28).

[CFJ03b] H. Chen, T. Finin, A. Joshi. “An Ontology for Context-Aware Pervasive Computing Environments.” In: *The Knowledge Engineering Review* 18 (2003), pp. 197–207. DOI: 10.1017/S0269888904000025 (cit. on p. 32).

[Dam07] N. Damij. In: *Business Process Management Journal* 13 (2007), pp. 70–90. DOI: 10.1108/14637150710721131 (cit. on p. 18).

[Dis15] M. Distefano. *Cloud Computing and the Internet of Things: Service Architectures for Data Analysis and Management*. 2015 (cit. on p. 17).

[DPK+09] M. D. Dikaiakos, G. Pallis, D. Katsaros, P. Mehra, A. Vakali. “Cloud Computing: Distributed Internet Computing for IT and Specific Research.” In: *IEEE Internet Computing* 13 (2009) (cit. on p. 18).

[DR07] S. Dustdar, F. Rosenberg. “A Survey on context-aware systems.” In: *International Journal of Ad Hoc and Ubiquitous Computing* (2007) (cit. on p. 22).

[DS17] J. Dörndorfer, C. Seel. “A Meta Model Based Extension of BPMN 2.0 for Mobile Context Sensitive Business Processes and Applications.” In: *Proceedings of the 13. Internationale Tagung Wirtschaftsinformatik* (2017) (cit. on p. 20).

[EBF+17] C. Endres, U. Breitenbücher, M. Falkenthal, O. Kopp, F. Leymann, J. Wettinger. “Declarative vs. Imperative: Two Modeling Patterns for the Automated Deployment of Applications.” In: *Proceedings of the 9th International Conference on Pervasive Patterns and Applications (PATTERNS)*. Xpert Publishing Services, Feb. 2017, pp. 22–27. ISBN: 978-1-61208-534-0 (cit. on p. 39).

[End95] M. R. Endsley. “Toward a Theory of Situation Awareness in Dynamic Systems.” In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 37 (1995). DOI: 10.1518/001872095779049543 (cit. on p. 13).

[Fis06] L. Fischer. *Workflow Handbook*. Future Strategies, Inc, 2006 (cit. on p. 18).

[GFHK14] H. Guermah, T. Fissaa, M. N. Hatim Hafiddi, A. Kriouile. “An Ontology Oriented Architecture for Context Aware Services Adaptation.” In: *International Journal of Computer Science Issues* 11 (2014) (cit. on pp. 25, 28, 31, 32).

[GO14] L. González, G. Ortiz. “An Event-Driven Integration Platform for Context-Aware Web Services.” In: *Universal Computer Science* 4 (2014), pp. 1071–1088 (cit. on p. 76).

[GWPZ04] T. Gu, X. H. Wang, H. K. Pung, D. Q. Zhang. “An Ontology-based Context Model in Intelligent Environments.” In: *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference* (2004) (cit. on p. 32).

[KBBL12] O. Kopp, T. Binz, U. Breitenbücher, F. Leymann. “BPMN4TOSCA: A Domain-Specific Language to Model Management Plans for Composite Applications.” In: *Business Process Model and Notation*. Ed. by J. Mendling, M. Weidlich. Vol. 125. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2012, pp. 38–52. ISBN: 978-3-642-33154-1. DOI: 10.1007/978-3-642-33155-8_4 (cit. on p. 37).

[KBS+16] K. Képes, U. Breitenbücher, S. G. Sáez, J. Guth, F. Leymann, M. Wieland. “Situation-Aware Execution and Dynamic Adaptation of Traditional Workflow Models.” In: *Proceedings of the 5th European Conference on Service-Oriented and Cloud Computing (ESOCC)*. Springer International Publishing, 2016, pp. 69–83. DOI: 10.1007/978-3-319-44482-6_5 (cit. on pp. 13, 41).

[KKR+13] M. Knappmeyer, S. L. Kiani, E. S. Reetz, N. Baker, R. Tonjes. "Survey of Context Provisioning Middleware." In: *IEEE COMMUNICATIONS SURVEYS TUTORIALS* 15 (2013) (cit. on p. 22).

[McC93] J. McCarthy. "Notes on formalizing context." In: *Proceeding IJCAI'93 Proceedings of the 13th international joint conference on Artificial Intelligence* 1 (1993), pp. 555–560 (cit. on p. 56).

[MG11] P. Mell, T. Grance. "The NIST Definition of Cloud Computing." In: *NIST Special Publication* (2011) (cit. on p. 17).

[NH13] D. Novakovic, C. Huemer. "A Survey on Business Context." In: *Intelligent Computing, Networking, and Informatics. Advances in Intelligent Systems and Computing* 243 (2013). DOI: 10.1007/978-81-322-1665-0_19 (cit. on pp. 25, 28, 31, 32).

[NPB13] N. Z. Naqvi, D. Preuveneers, Y. Berbers. *Cloud Computing: A Mobile Context-Awareness Perspective*. Springer London, 2013. ISBN: 978-1-4471-5106-7. DOI: 10.1007/978-1-4471-5107-4_8 (cit. on p. 17).

[PFCP10] M. Poveda, M. C. S. Figueroa, R. G. Castro, A. G. Perez. "A Context Ontology for Mobile Environments." In: *Proceedings of Workshop on Context, Information and Ontologies - CIAO 2010 Co-located with EKAW 2010* 626 (2010) (cit. on p. 21).

[RRF08] M. Rosemann, J. Recker, C. Flender. "Contextualization of Business Processes." In: *International Journal of Business Process Integration and Management* (2008) (cit. on p. 42).

[Saa16] K. Saatkamp. *Modeling Approaches to Enable Data Shipping and Function Shipping by Means of TOSCA*. 2016 (cit. on p. 77).

[Sat01] M. Satyanarayanan. "Pervasive Computing: Vision and Challenges." In: *IEEE Personal Communications* (2001) (cit. on p. 13).

[Sil11] B. Silver. *BMPN Method  Style With BPMN Implementer's Guide*. Cody-Cassidy Press, 2011 (cit. on p. 18).

[SL04] T. Strang, C. Linnhoff-Popien. "A Context Modeling Survey." In: *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*. 2004 (cit. on pp. 29, 30).

[SML01] M. Samulowitz, F. Michahelles, C. Linnhoff-Popien. "CAPEUS: An Architecture for Context-Aware Selection and Execution of Services." In: *IFIP International Federation for Information Processing* 70 (2001). DOI: 10.1007/0-306-47005-5_3 (cit. on pp. 24, 31, 32).

[SXV] O. Sofela, L. Xu, P. T. D. Vrieze. “Context-Aware Process Modelling through Imperative and Declarative Approach.” In: *FIP Advances in Information and Communication Technology* 408 (), pp. 191–200. DOI: 10.1007/978-3-642-40543-3_21 (cit. on p. 38).

[TJH10] C. Timmerer, J. Jabornig, H. Hellwagner. “A Survey on Delivery Context Description Formats – A Comparison and Mapping Model.” In: *Journal of Digital Information Management* 8 (2010) (cit. on p. 27).

[WB10] Y. Wei, M. B. Blake. “Service-Oriented Computing and Cloud Computing.” In: *IEEE Computer Society* (2010) (cit. on p. 18).

[WGZP04] X. H. Wang, T. Gu, D. Q. Zhang, H. K. Pung. “Ontology Based Context Modeling and Reasoning using OWL.” In: *PERCOMW ’04 Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops* (2004) (cit. on p. 32).

[YBSD16] A. Yousfi, C. Bauer, R. Saidi, A. K. Dey. “uBPMN: A BPMN extension for modeling ubiquitous business processes.” In: *Information and Software Technology* 74 (2016), pp. 55–68 (cit. on p. 21).

[YD16] D. S. Yadav, P. K. Doke. “Mobile Cloud Computing Issues and Solution Framework.” In: *International Research Journal of Engineering and Technology (IRJET)* 03 (2016) (cit. on p. 18).

[YMCL13] P. Yu, X. Ma, J. Cao, J. Lu. “Application mobility in pervasive computing: A survey.” In: *Pervasive and Mobile Computing* 9 (2013), pp. 2–17 (cit. on p. 13).

All links were last followed on Juli 19, 2017.

**Declaration**

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature