Multi-Dimensional Context-Aware Adaptation of Service Front-Ends

Project no. FP7 – ICT – 258030

Deliverable 3.1.2 Reference Models Specification (R2)

Due date of deliverable: 31/08/2012

Actual submission to EC date: 30/08/2012

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination level		
[PU]	**[Public]**	**Yes**

Document Information	
Lead Contractor	UCL
Editor	Vivian Genaro Motti
Revision	ISTI
Reviewer 1	
Reviewer 2	
Approved by	
Project Officer	Michel Lacroix

Contributors	
Partner	**Contributors**
UCL	Vivian Genaro Motti, Nesrine Mezhoudi, Sophie Dupuis, Pascal Beaujeant, Jean Vanderdonckt

Changes			
Version	**Date**	**Author**	**Comments**
1	24/06/2012	UCL	Basic structure of the document: table of content, initial content based in the Description of work
2	10/08/2012	UCL	Contents, Review 1
3	22/08/2012	ISTI	Internal review
4	28/08/2012	UCL	Final Contents

Executive Summary

The main goal of task 3.1 is to define reference models to support developers in the implementation of applications that perform context-aware adaptation. The models formalize and define concepts that are relevant for the adaptation process as well as their specific properties and relationships.

The elaboration of UML diagrams provides a graphical visualization of adaptation concepts from different perspectives, for example from the user perspective, the tasks that can be performed, and from the system perspective, the states of execution. The diagrams also provide a unified view of Serenoa and its essential concepts regarding the adaptation process.

In the first release the models are defined in a high abstraction level, generic enough to allow them to be used in a wide range of domains for adaptive and adaptable applications and to accommodate future changes along the evolution of the project. The second release of this deliverable complement the first one, by presenting a meta-model of context-aware adaptation specified in more detail, according to the evolution of the architecture of Serenoa and focusing on application scenarios.

Table of Contents

1 Introduction

1.1 Objectives

This deliverable is dedicated to investigate and to create reference models for context-aware adaptation (CAA). Such models aim at formalizing and standardizing concepts, which are essential to define, to implement and to execute context-aware adaptation. In the first release of this deliverable we focused in defining 5 reference models for CAA, namely: Use Case Diagrams, Sequence Diagrams, Class Diagram, Statechart and a Meta-model. All definitions concern a high abstraction level, so as to be generic enough to comprehend all adaptation types (i.e. adaptivity and adaptability), steps, to accommodate different application domains and scenarios and also to enable further refinements. The second release of this deliverable extends and refines the previous meta-model presenting a more complete and extended version of it.

1.2 Audience

The target audience consists of researchers and practitioners with interest in modelling the adaptation process.

1.3 Related documents

- The Deliverable 1.2.1 - Architectural Specifications defines the entities used to structure the life lines of the Sequence Diagrams and requirements that must be respected.
- The Deliverable 2.1.1 - CARF and CADS contains the catalogue of adaptation techniques, dimensions and concepts that aid the composition of the Reference Models (in particular the Use Cases)
- The Deliverable 2.2.1 - CARFO (R1) is related with and complements this document once both documents formally represent the concepts of adaptation, mainly the ones previously and partially defined by the CARF and CADS
- The Deliverable 3.1.1 – Reference Models (R1) presents the first version of all reference models for CAA
- The Deliverable 3.3.1 – AAL-DL: Semantics, Syntaxes and Stylistics, which describes the requirements for the language, provided a set of principles that the models must also be aligned with.
- The Deliverable 4.2.1 - Algorithm Library for AAL details and complements the flows (basic and alternative) presented in the descriptions of the Use Case diagrams

1.4 Organization of this document

Chapter 1 presents the objectives, audience and related documents with this deliverable. In Chapter 2, the description of work is presented and illustrated with examples. Chapter 3 describes related work. Chapter 4 specifies the Meta-model for CAA. Chapter 5 presents final remarks and concludes this deliverable.

2 Description of Work

2.1 Motivation

The creation of models aims at structuring, organizing and formalizing the relevant concepts that support the development of adaptive and adaptable systems in a standard fashion. Models can be presented by a natural language or some abstract syntax using literals and (or) non-literals. This syntax determines whether the model is also meaningful to the machine, e.g. models expressed in natural language vs. logic-based models [Gomez and Tran, 2009].

Model-driven engineering seems to be one of the central methods that connect model-based interface development with software engineering. Mainly because of the concepts they share: models that describe different aspects of an interactive system [Luyten et al., 2010].

Software engineers are primarily concerned with choosing appropriate models to represent specific application domains. The use of a meta-model not only guarantees a strong and focused semantics tied to a particular application domain, but also offers a precise abstract syntax and a common representation to any developed model [Farias et al., 2007]. By defining meta-models, we provide a standard abstraction to specify CAA in terms of rules, constraints, relationships, and other theoretical concepts that are closely related to context-aware adaptation supporting its implementation.

According to Simons (2007), visual languages play an important role in software engineering because graphical models are better readable and understandable by human beings.

2.2 Goal

A model is a simplification of the reality that provides the blueprints of a system, encompassing detailed or general plans [Booch et al., 1998]. The definition of reference models intends to provide support for developers and designers to implement adaptation. The main goal is to create models that formally define the knowledge about adaptation in order to help the development of adaptive and adaptable applications. The models help the development of these applications by formally defining the knowledge about the concepts, properties and relationships involved with adaptation. In addition to this, the adoption of UML helps us to [Booch et al., 1998]:

- Visualize the system to be
- Specify the structure and behavior of the system
- Define templates to guide the implementation of the system
- Document important decisions

By modeling the related concepts, it is possible to provide support for design decisions at an early stage of the development life cycle and consequently to identify potential critical points that may appear during it. This approach also intends to optimize all the steps of a development phase.

2.3 Description

Task 3.1 Reference Models for CAA of SFEs

This task consists in creating a set of Reference Models (RMs) that enables and support the implementation of context-aware SFEs in accordance with the CARF produced in Task 2.1.

The RMs will be concerned with the conceptualization and formal representation of:

- The enriched context of use (Environment, Platform, User, UI Services, Situations, …).
- The different layers of abstraction that are found in the development of UIs (Task and Domain, Abstract and Concrete User Interface).

The RMs will be specified primarily by means of MDA standards (UML 2 and MOF).

3 Modelling Context-aware Adaptation

This section presents a set of related work that motivated, and inspired the definition and refinements of the reference models.

3.1 Justification

Context-aware adaptation combines activities of gathering the context and performing adaptation (normally through rules). To do so, it is necessary to formally represent both, context and adaptation rules. Usually context and adaptation rules are either directly implemented or defined by intermediate models that can be further processed to perform adaptation [Gomez and Tran, 2009]. Given that adaptation can be applied in many different domains, modelling raises as a solution that unifies its representation, assuring mainly interoperability, reuse and consistency.

3.2 Meta-Models for Context-Awareness and Adaptation

Several authors have been investigating the modelling of context-aware adaptation or closely related concepts. In this section we present relevant contributions in this domain and discuss their concepts that are important for performing adaptation. We also discuss their strengths and weaknesses.

The context information, as mentioned in the Deliverable 3.1.1, can be modelled in many different ways: by key-value pairs, markup languages (e.g. XML), ontologies (e.g. OWL + RDF), modelling languages (e.g. UML), etc [Strang and Linnhoff-Popien, 2004], [Farias et al., 2007]. Besides this, the context information includes an extensive set of concepts, mainly belonging to Users, Platforms and Environments. Therefore, there is also an extensive list of works dedicated to formalize context. This section provides a brief overview about 8 works, partially illustrates their models, and discusses their strengths and weaknesses. The works presented below are organized by chronological order.

3.2.1 A Meta-Model Approach for Context Information (by Fuchs et al., 2005)

According to Fuchs et al. (2005) a model unifying the representation of context facilitates the exchange of information, provides a common understanding and enables interoperability. To model context with an ontology, the authors considered specific characteristics, such as quality levels and rules. Reasoning, inferences and extensibility were also considered.

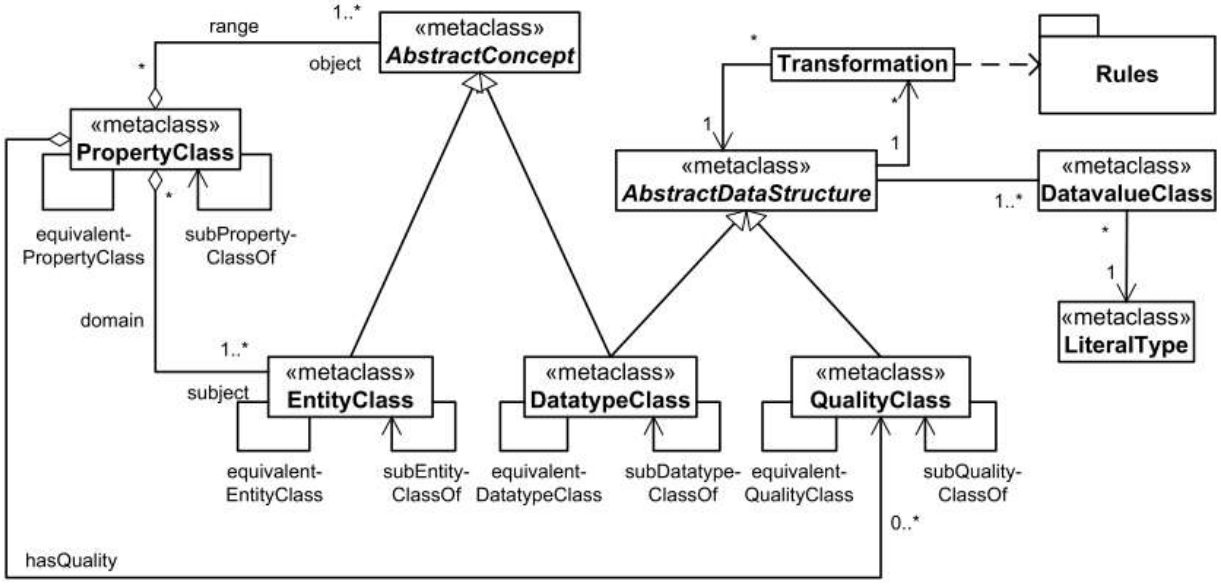

Figure 1. A meta-model layer illustrating its constructs (by Fuchs et al., 2005)

Three classes, as Figure 1 illustrates, work as base constructs for representing context knowledge:

- **Entity:** represents a group of entities that share the properties in common;

- **Datatype:** represents a type of data (e.g. temperature, noise level, position);
- **Property:** represents a relationship between an instance and another one or a datatype;
- **Quality:** represents a specific quality (e.g. precision, resolution, time or certainty).

The dependencies between concepts that define specific transformations are expressed as rules, implying in an antecedent and consequent relation of the conditions.

To better specify data types, the authors adopted two special constructs, namely:

- **Datavalue class:** to specify data structures;
- **Transformation:** to define a conversion function.

As important principles that must orient the adaptation, four were highlighted: the consistency (aiming at checking the validity of the meta-model), reuse, extension and interoperability.

Fuchs et al. (2005) states that the main advantages on adopting models for describing context consists in the reuse of existing consensus, which results in higher quality models and also in a higher interoperability. They proposed a meta-model for context information taking into account its constraints, requirements, qualities and certain principles, such as: interoperability and formality.

3.2.2 A Meta-Model Diagram for Context of Use

This work, a result of the NEXOF-RA Investigation Team on Declarative UI Authoring and Context Model, is inspired by the Cameleon Reference Framework, and it is composed by a set of 8 models, namely: context meta-model, context-of-use meta-model, domain, task, abstract user interface, concrete user interface, behaviour, and mapping models. In this section we focus on the context of use meta-model created to capture the context of use in which a user is interacting with a specific computing platform in a given physical environment.

This meta-model, as Figure 2 illustrates, is composed by 8 entities and 5 associations. The entities are:

- **Context of Use:** represents the whole context of use.
- **Environment:** models the physical environment where the interaction takes place.
- **User:** models a human being that is interacting or perceiving the system in a given moment. It has one attribute named *att*.
- **Delivery Context:** it encompasses the set of hardware, software or a network platform, i.e. the computational, interaction or communication components used to perceive and interact with a given system. It also has one attribute named *att*.
- **Aspect:** represents a class of a hardware or a software, it is a resource that can be part of the delivery context, and it participates in delivering a user experience. For example: a camera, a device, or a browser. It also has one attribute, named *att*.
- **Component:** is an instance of an aspect. An aspect, in any delivery context, can have zero or more components. For example a device with two or three cameras. It also has one attribute named *att*.
- **Characteristic:** models a context entity representing a featured supported by a delivery context component. Examples include: image formats, such as gif and jpeg, markup languages, or font types. It has as an attribute the name of the characteristic.
- **DDR:** represents a Device Description Repository, a specialization of a context provider responsible for gathering and maintaining information about characteristics and behaviours of components of a delivery context.

And the association types considered are:

- **Active:** links a delivery context with active components, i.e. components that are actually participating in delivering the user experience;
- **Available:** links a delivery context with components that are ready to be activated by the end user;
- **Component:** links a delivery context with its components;
- **Default:** links a delivery context with components that will be used if no preference of the user is set;
- **Supports:** links a component of the delivery context with its characteristics.

By analysing such a meta-model we observe that the context of use includes explicitly as context dimensions the user, the environment and the delivery context, and it is composed by context elements and properties.

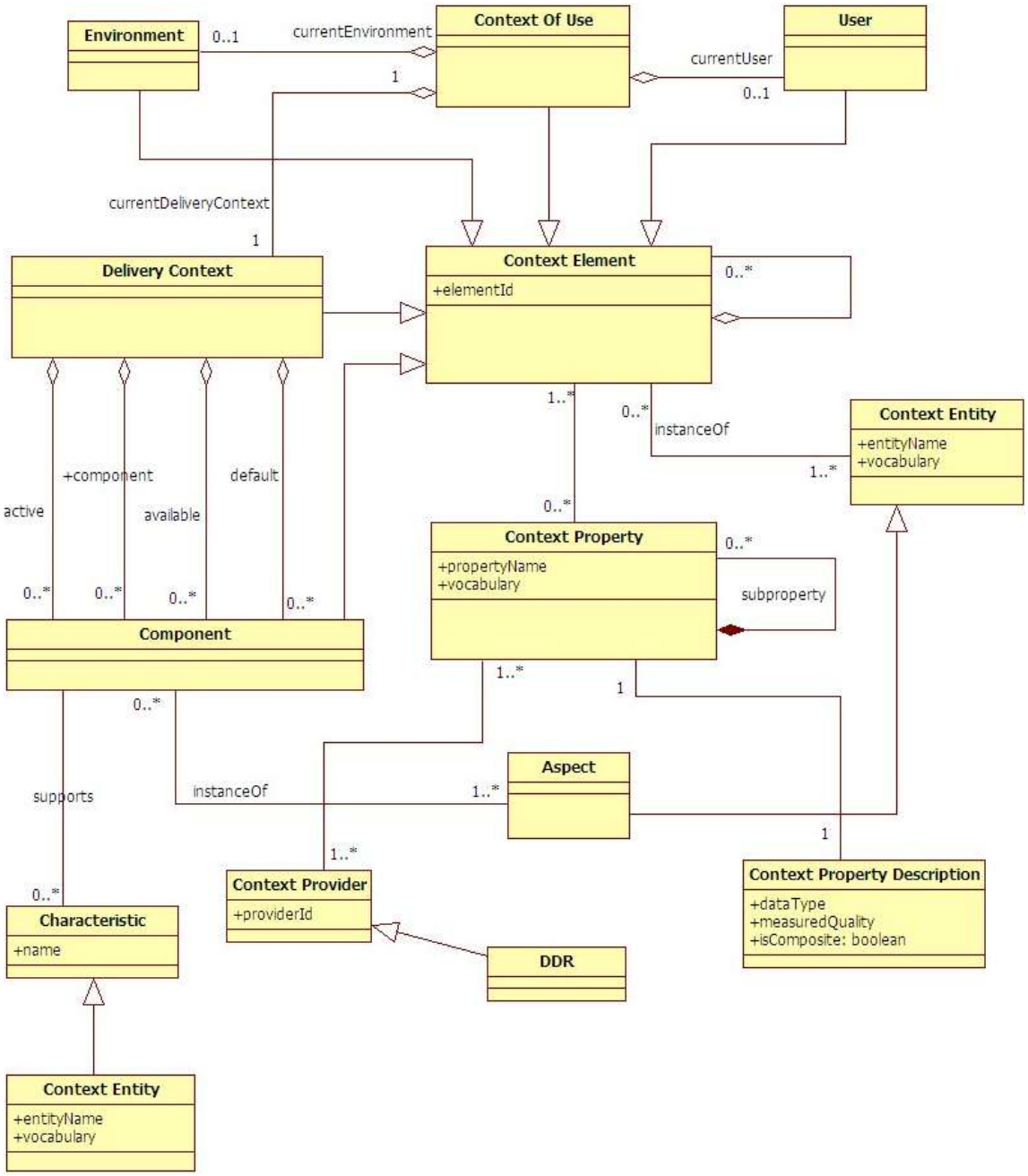

Figure 2. Context of Use Meta-model Diagram (by MORFEO Project)

3.2.3 UsiXML Context Meta-Model

UsiXML supports an MDE approach by defining a set of meta-models that contain all elements and relations included in the different models. It tries to cover all models that are required for user interface analysis and design, and encompasses models that are commonly used in model-based interface development.

Figure 2 the target is the context of use. From literature, context has a vague definition that encompasses several elements. The UsiXML context model selects relevant concepts for a model-driven engineering process. However, context is a dynamic entity, therefore its models are usually subject to continuous changes [Luyten et al., 2010].

In the meta-model illustrated in Figure 3, mainly the platform is taken into account, information considered include: the type of hardware (e.g. colors, sound output, text input, touch screen, keyboard), the network

characteristics (e.g. capacity), browser type (e.g.name, version, html support), and the software type (e.g. handwriting recognition, and audio input encoder).

3.2.4 A UML Model for Context profile (by Simons 2007)

The Context Modelling Profile (CMP) is a lightweight UML extension for context models targeted at mobile distributed systems. This model formalizes meta-information of the context, i.e. source and validity of context information, and reflects privacy restrictions. The profile provides several well-formed rules for context models supporting the development of context-aware mobile applications through visual modelling language.

In the development process of a context-aware application that should adapt to the user context, the developer has to define which context information are relevant for the adaption of the application [Simons, 2007].

A context model must reflect meta-information about it, e.g. if it is a dynamic or static data (like the location and the birthday of the user).

Simon (2007) defined a meta-model for specifying context information (Figure 3). It is composed by several classes, such as the stereotype *ContextItem*, which must be applied to classes that represent a specific context item. A *ContextAssociation* is used for technical purposes to restrict associations between context item types.

CMP restricts the use of UML, adapting it to context modelling purposed. CMP can be integrated in many UML modelling tools. To model dynamic behaviour, i.e. contextual situations and adaption strategies, which are necessary to model user triggered and automatic adaption of context-aware applications, further stereotypes must be added to the profile.

Simons (2007) believes that models resulting from the application of CMP are still easy to understand even when dealing with more complex scenarios.

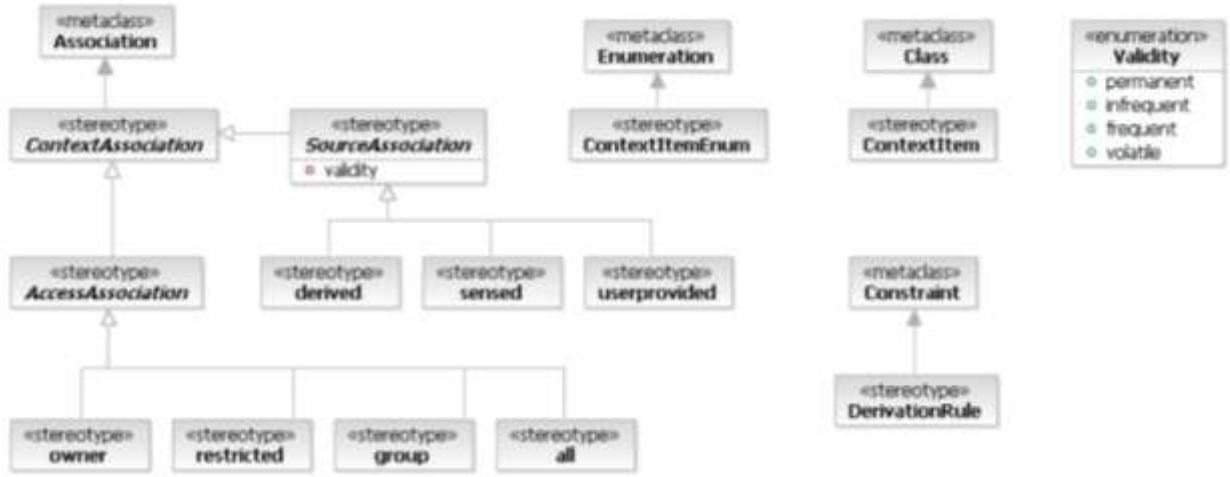

Figure 3. The Context UML profile (by Simons, 2007)

3.2.5 A Meta-Model for Adaptation Rules (by Ganneau et al., 2007)

Plasticity is a principle that ensures that a UI is able to appropriately adapt itself to its context [Ganneau et al., 2007]. Based on this principle, and in the CARE properties of complementarity, assignment, redundancy and equivalence, Ganneau et al. (2007) created a meta-model for adaptation rules. It includes the context of use, an adaptation engine and the user model.

In a context of pervasive computing, the plasticity of UI becomes inevitable. However, although the UIs must change according to different contexts of use during the adaptation, both functional and non-functional requirements must be properly preserved. The meta-model proposed by Ganneau et al. (2007) is a descriptive and generative formalization of adaptation rules, that conceptually aids design decisions and supports in practice the implementation of CAA. The authors define adaptation in three main steps: gathering the context, taking decisions about adaptation and executing it. In this context they emphasize that the users'

preferences must always have the highest priorities.

The adaptation engine defines ECA-rules in which given a specific event, the condition is evaluated and, if valid, the action executed. The event can be a change in the context, or in the system itself. The user model aims at assuring comfort, efficiency and safety to the end user. It collects the user choices, infers their preferences and creates new adaptation rules if necessary.

In this scenario, this meta-model composed by 9 classes and 3 enumerations, as Figure 4 illustrates, aims at supporting stakeholders in their design decisions and at facilitating the implementation of adaptation.

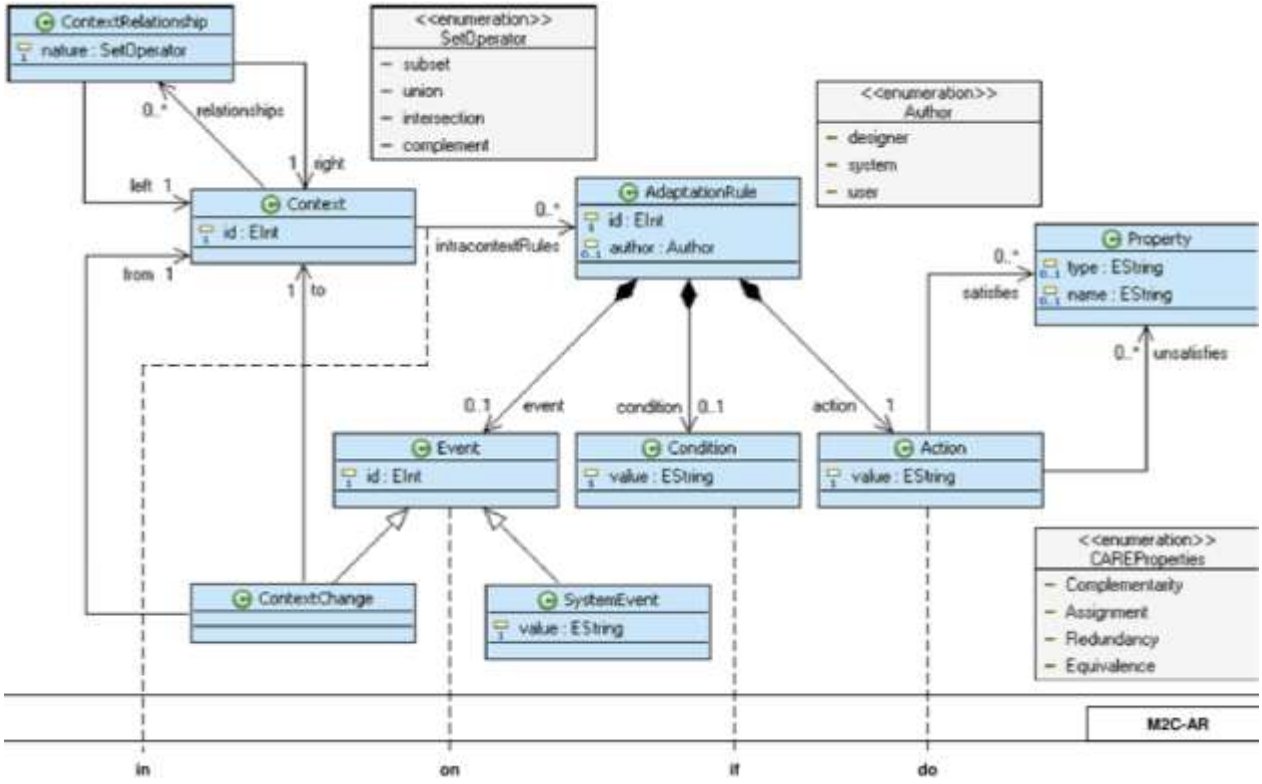

Figure 4. A meta-model for adaptation rules (by Ganneau et al., 2007)

3.2.6 A Meta-Model for Context-aware Mobile Applications (by Farias et al., 2007)

Targeting at the development of mobile application, Farias et al. (2007) state that a common context meta-model is beneficial for defining both: applications and platforms. In their work, the OMG Meta Object Facility [MOF] was used in the creation of a meta-model. It is based on three fundamental concepts, namely entities, attributes and associations and it is structured in 7 main components:

- **Context Interpreter:** manipulates and refines the context information, provides mechanisms for context acquisition and manipulation;
- **Context Provider:** provides contextual information to the platform (e.g. sensors);
- **Service Provider:** is an entity that provides a service;
- **Context-Aware Application:** applications that interact with the platform trough services;
- **Service Manager:** supports the use of platform services being responsible for their publication, discovery, selection, composition and execution;
- **Registry Manager:** provides a common interface for accessing platform records, such as: services, users' profiles, context history;
- **Monitor:** interprets and manages subscriptions and triggers services.

In Figure 5 the class entity corresponds to physical or conceptual objects that capture context information. This information can be classified as *static* (for long term values) or *dynamic* (for temporal values) depending on its *association* type. Association defines relationships between *entities*, and *attributes*.

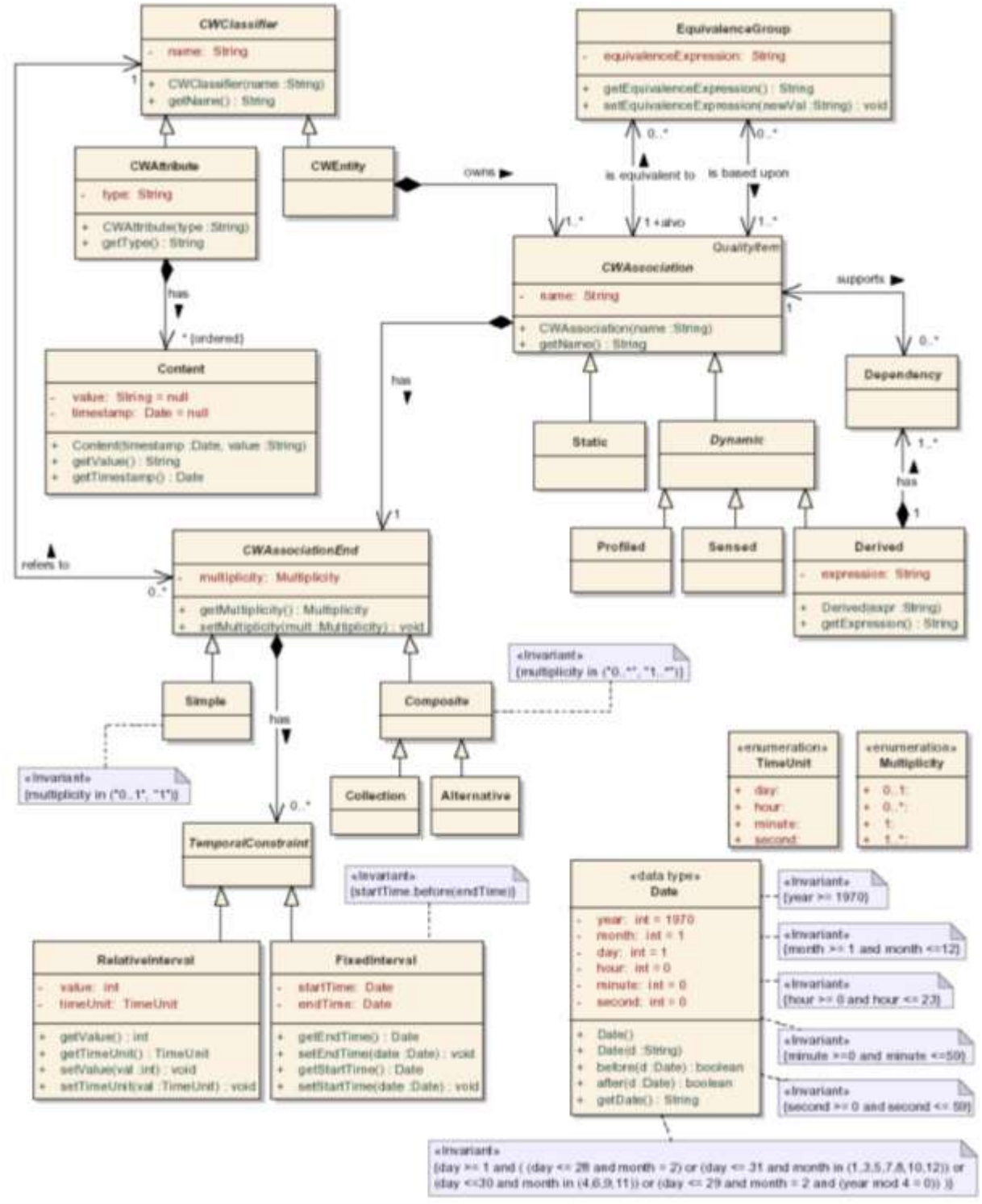

Figure 5. A core view of a context meta-model for mobile applications (by Farias et al., 2007)

Three association types (*sensed*, *profiled* or *derived*) were defined stressing differences of origin, persistence, confidence level and temporality.

The concepts of this context meta-model were identified and grouped according to 5 views, each one represented by a MOF package, viz., core (Figure 6), service, subscription, context-aware service and quality views.

OCL was employed to refine the semantics of the meta-model by means of invariants, as illustrated in Figure

5.

As important principles, the authors highlight: loose coupling, flexibility, adaptability and platform-independency. To assure such requirements, and interoperability, a service-oriented architecture was adopted.

Services can be composed by other services, must have a provider (entity responsible for providing the service), the entity *ContextSourceService* provides primary contextual information to any platform. A service consists of operations, messages, and their parts (*classifier*, *entity*, or *attribute*).

This meta-model provides a formal representation of context, enabling the use of different tools to develop and to manipulate models. Context-aware mobile application can benefit from the meta-model by re-using its solutions. Although interoperability is assured, so far there are no mappings corresponding platform-independent to platform-specific definitions.

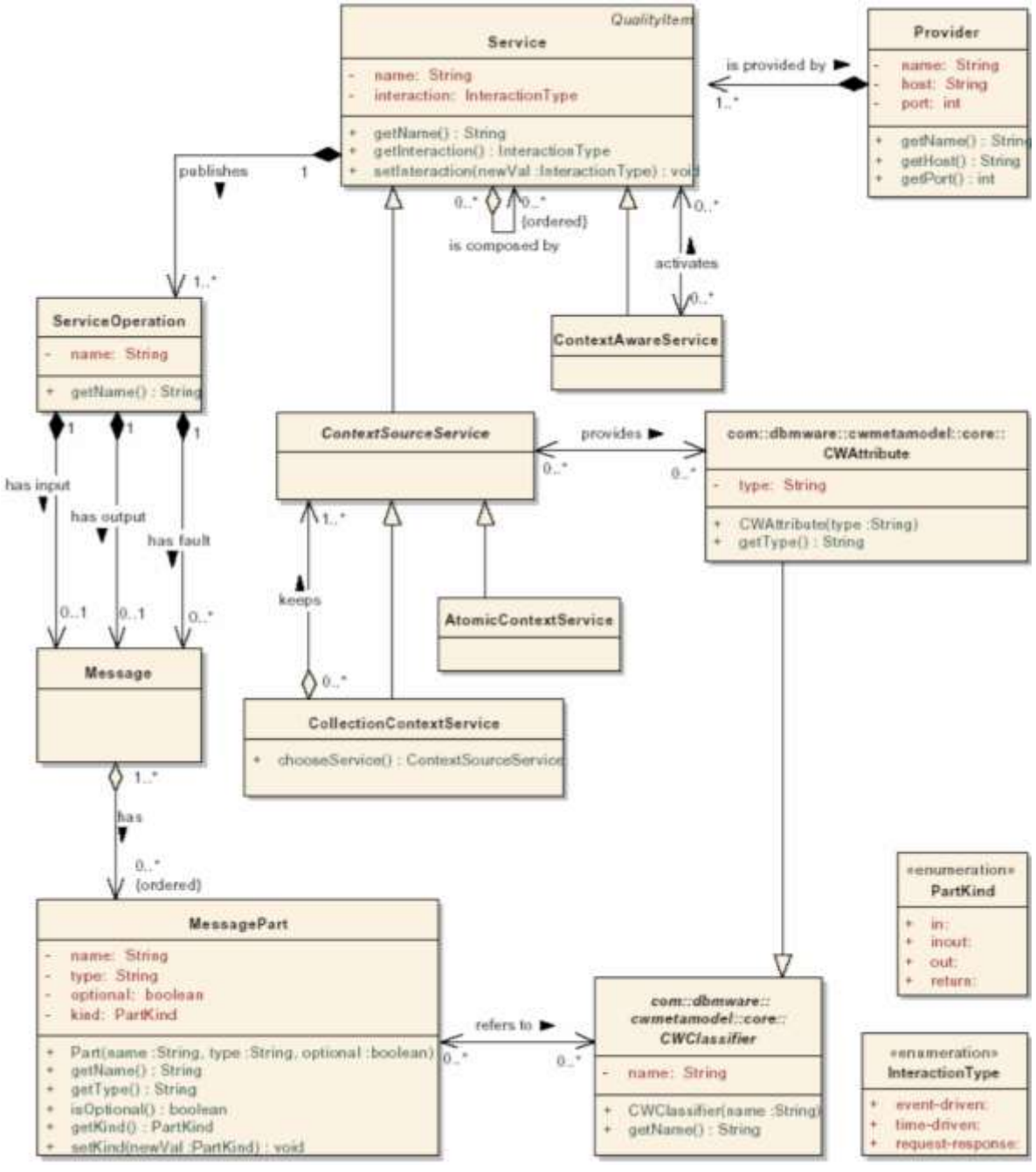

Figure 6. A service view of the context meta-model, it assures decoupling of applications (by Farias et al., 2007)

3.2.7 A Meta-Model for Adaptation Rules (by Lopez et al., 2009)

Aiming to simplify the development of applications that accommodate ubiquitous interaction, multiple devices and a heterogeneous mass of users, a model-based approach is recommended [Lopez et al., 2009]. With a systematic approach, models can be transformed in code to be executed by the end user. Besides this, to adapt the UI for different contexts of use, a set of rules is required. Dynamic rules for instance can be properly adjusted to evolve according to the dynamic changes of the context.

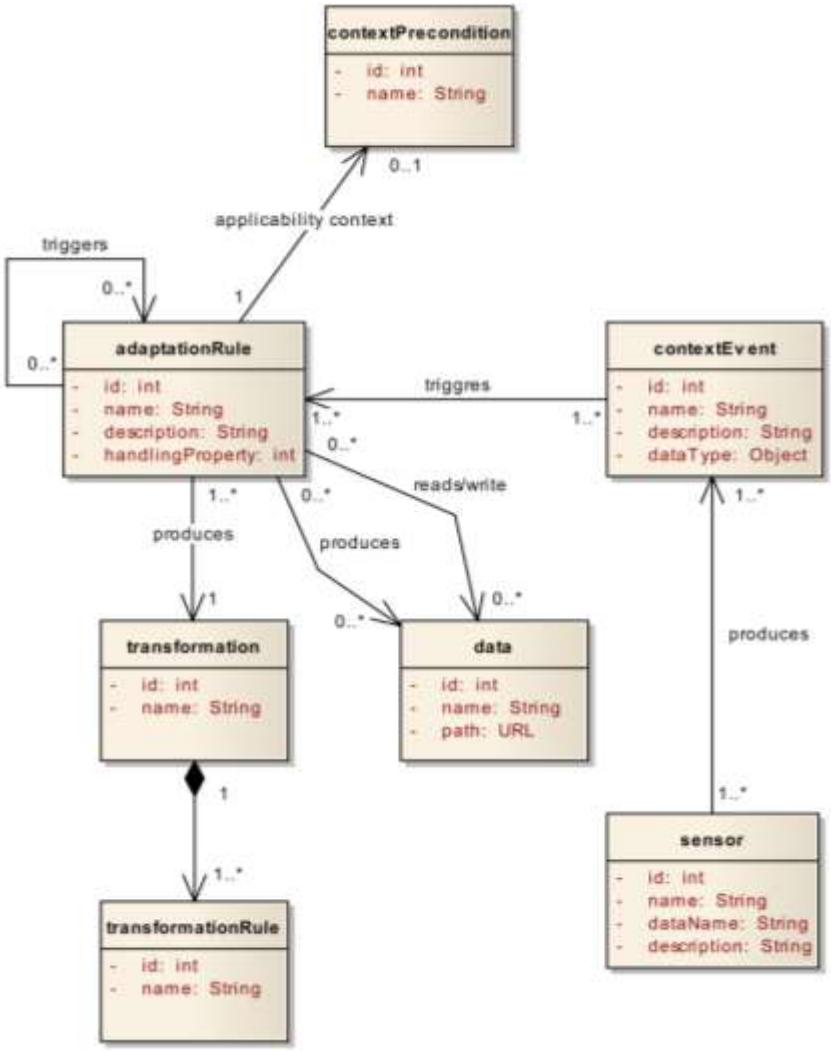

Figure 7. A Meta-model formalizing Adaptation Rules (by Lopez et al., 2009)

For Lopez et al. (2009), given the heterogeneous scenarios of interaction, adaptation rules should be a must in UIs' development practices. According to them, a meta-model not only aids to standardize adaptation but also enables the reuse of adaptation rules.

Figure 7 depicts a meta-model for adaptation rules. It models: events that are related to the context and that trigger the rules, sensors, that are responsible for producing events, data, that are accessible by the rules, as well as transformation specifications, and context preconditions.

According to Lopez et al. (2009) a model must support the transformation between models, and users must always be able to manipulate and to customize the adaptation.

3.2.8 A Meta-Model for the Context of the User (by Kaklanis et al., 2010)

Aiming a design for all, and considering impairments in hearing, vision, mobility, and strength that are common to the aging process, Kaklanis et al. (2010) created a virtual user model. It is a holistic framework that supports accessibility testing of novel products and applications. The authors consider physical, cognitive, behavioural and psychological aspects of end users, in a dynamic and flexible user model.

The virtual user model, as Figure 8 partially illustrates, is a complex model expressed using UsiXML format and includes users' preferences, needs and capabilities. The classes defined include: gender, name, age group, speech, general preferences and behaviour.

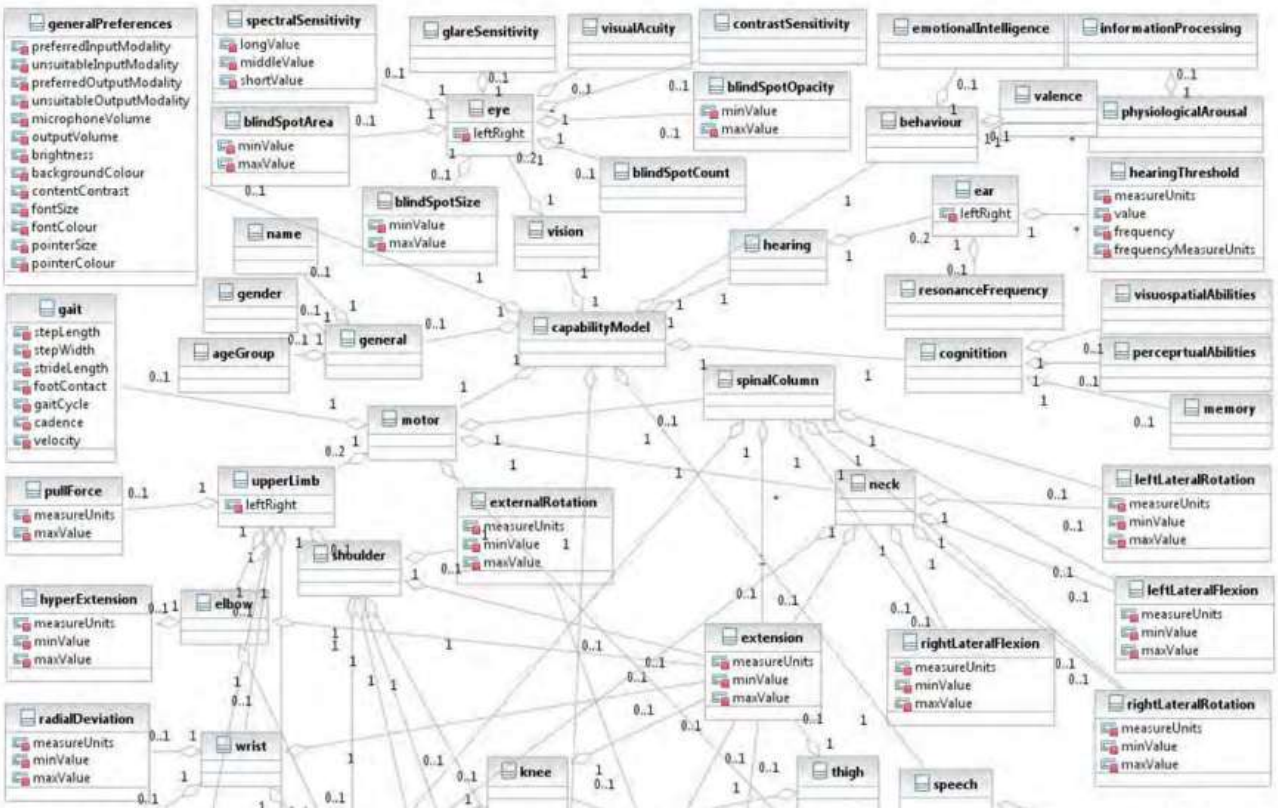

Figure 8. A User Model (partial view), by Kaklanis et al. (2010)

There are also other models created in order to specify the user. Duran et al. (2010) created a meta-model for specifying uses in the domain of context-aware recommender systems. They believe it is important to share users' information in a standard manner, and use such information to offer them personalized contents and services.

3.2.9 Discussion

In the sections above, we illustrated and briefly described 8 related works that targeted at modelling adaptation. By the analysis of such works we could derive some shortcomings and requirements that together with Serenoa goals lead to the definition of a meta-model for context-aware adaptation.

Most of the related works that are associated with modelling context-aware adaptation (or its specific properties) are focused on specific dimensions of this domain.

For instance, the works of Fuchs et al. (2005), MORFEO and UsiXML are targeted at modelling context information and its dimensions. The work of Kaklanis et al. (2010) is specific to modelling the context of end users. No explicit mention is done to adaptation or rules. Simons (2007) and Farias et al. (2007) are also targeted at modelling context information, but their focus is in the domain of mobile applications. The works of Ganneau et al. (2007) and Lopez et al. (2009) are also targeted at one specific concept of adaptation, being both dedicated at modelling adaptation rules.

Although all of the works analysed provide relevant contributions to specific concepts within the domain of context-aware adaptation, we lack a model that is capable of integrating the main concepts involved, i.e. context, adaptation rules and the UI model generation.

Moreover, we could observe that when the works are targeted at modelling the context of use, not always the context information is considered in a broad perspective, i.e. including its three main dimensions, of users, platforms and environments jointly, and their respective properties. Besides this, sometimes a specific

application domain is emphasized, which could pose some constraints for context-aware adaptations that target at more generic or different domains.

In order to summarize the analysis of the related works presented above we elaborated a table in which we explicitly mark the context targeted and the application domain of interest. Table 1 illustrates the main focus of such works concerning the context dimensions considered (i.e. users, platform or environment), and the application domain of interest. The principles envisaged, if any, are also specified.

Table 1. Analysis of the related works regarding context information dimensions and domain of interest

Author	Context			Application Domain	Principles
	User	Platform	Environment		
[Fuchs et al., 2005]	X		X		Consistency, formality, reuse, extensibility, interoperability
MORFEO	X	X	X		
UsiXML	X	X			
[Simons, 2007]		X		Distributed mobile systems	
[Ganneau et al., 2007]	X			Pervasive Computing	Plasticity
[Farias et al., 2007]		X		Mobile-Applications	
[Lopez et al., 2009]		X			
[Kaklanis et al., 2010]	X				Accessibility

By analysing the Table 1, we observe that in the scope of the related works analysed, they mainly target at modelling the platform and the users as context dimensions, while the environment, on the other hand, is rarely (explicitly) considered. Mobile applications are also commonly targeted by such models.

As important principles, that guide the modelling of context or adaptation concepts, we can mainly highlight: consistency, formality, reuse, extensibility, interoperability, plasticity and accessibility. Such principles are also aligned with the adaptation goals of Serenoa, as specified by: its (non functional) requirements (D1.1.1 and D1.1.2), the branch *why* of the CARF (D2.1.2) and the Evaluation Criteria (D2.4.2).

We believe that all dimensions of context are relevant, and the models by targeting a general-purpose approach can accommodate different application domains.

Based on the review and analysis of the related works and their main concepts, we created a meta-model for context-aware adaptation. Such meta-model is a refinement of the meta-model presented in the previous release of this deliverable (in D3.1.1), it accommodates different application domains and extends the current work in this field by adopting a broader perspective.

4 Context-aware Adaptation Models

We start this chapter by describing and presenting the meta-model for context-aware adaptation and then, following the same structure of the previous release of this deliverable, we present the remaining (UML) models, as the use cases for context-aware adaptation, emphasizing the updates performed.

4.1 Meta-Model

Based in the previous analysis of related works, the Serenoa requirements and its continuous progress, in order to formalize the most essential concepts that are necessary to implement and execute a CAA process, a meta-model for CAA was elaborated. It is illustrated by Figure 9. This meta-model, named CAMM, uses the OMG notation for UML Class diagrams, being associations represented by named lines (e.g., triggers), aggregations represented by open diamonds (e.g., resource property), and compositions represented by closed diamonds (e.g., User). This meta-model covers the complete adaptation process from context gathering to generation of final UIs presenting the resulting adaptation. CAMM abstracts the necessary concepts, establishes their relationships and defines some of their properties and parities. Besides this, additional information, such as constraints and cardinality of the relations, are also specified.

Four colours were applied in this meta-model in order to separate concepts belonging to distinct domains. Therefore, the classes represented by red blocks refer to the adaptation agents, the ones represented by green blocks refer to the context of use, the yellow blocks refer to the core of the adaptation process, and purple blocks refer to the model generation.

This MOF-based meta model diagram, as Figure 9 illustrates, shows with the red blocks three possible agents that can trigger an adaptation process: the system, the end user or a third party. These agents are abstracted as *Adapter*, as defined in the use case diagram too. Considering that an adaptation process may be composed by several phases, different agents can be responsible for taking decisions for each of them [Horvitz, 1999], characterizing a mixed-approach. For instance, the end user may start an adaptation process, and the system decides which is the most appropriate method among the available ones. Besides, the agent roles can be further refined according to their specific characteristics and interrelationships, which permits collaboration and hierarchies.

A CAA process can also be triggered by a change in the context of use, as proposed by Ganneau et al., 2003. The green blocks in the meta-model diagram represent concepts related to the context information. The context defines the adaptation rules since it provides information to instantiate them. For instance, when the user changes the orientation of the device, a technique like ‘change the UI orientation’ must be applied, rotating the content of the UI according to the new device position (information gathered for instance by a sensor). We opt to specify context by its abstract dimensions of User, Platform, Environment and Application. Surely for each of these dimensions there is an extensive set of concepts involved. For sake of simplicity and readability, we refer to the sub-concepts of CARF to specify such dimensions (branch *to what*), to the CARFO, and further related works (e.g. Kaklanis et al. (2010)).

As the context is a composition of information gathered from different dimensions, there are sets of rules that can be simultaneously applied. An adaptation process is then governed by one or more rule. Rules, represented in the meta-model diagram by the yellow blocks, can be syntactically structured in the form of ECA rules (event, condition and actions) [Dittrich, 1995], instantiated and triggered by context information. The definition of adaptation rules in the meta-model diagram is inspired in the previous definitions as the ones proposed by Ganneau et al. (2007) and by Lopez et al. (2009). Both authors consider mainly: the context information, the transformation/action as a means of executing adaptation, and the events that may trigger the adaptation process.

Due to the fact that more than one rule is normally applied simultaneously, conflicts may appear. In order to solve them, priorities must be assigned for certain contexts: adaptation techniques may be abstracted in policies (meta-rules) that can also be further abstracted as strategies (meta meta-rules). An extension of ECA rules that includes also Justification can be applied as well. The compliance with ECA-rules, as well as the inheritance, nesting and prioritization of rules, are also stated as requirements for the AA-DL, as described in D3.3.1 Semantics, Syntax and Stylistics (R1) and in D1.2.1.

CAA results can be presented to the end user with different methods to prevent the end user disruption that is commonly caused by significant differences existing between the original UI and the adapted one. Animation is one possible method that can be applied in this sense. By using animation, the intermediary steps of a transition are explicitly presented to the end user, permitting her to intuitively comprehend sequential changes [Dessart, 2011].

As consequences of the actions performed by rules, models for SFE are generated. In the meta-model diagram, the models are represented by purple blocks. Following the principles of the model driven approach, the models range from task and concept level, abstract level, to concrete level and then final level [Calvary, 2003]. While a task model specifies the tasks and subtasks involved to accomplish a specific user goal, the final UI level specifies the layout issues (assuming a GUI), e.g. style, alignment, and colours. The support of different abstraction levels for UIs is also stated as a requirement for the AA-DL (as described in the D3.3.1).

The actions that are executed as a consequence of applying adaptation rules, can be implemented in terms of use cases (or adaptation techniques), as illustrates Chapter 4. The context is defined in alignment with the CARFO (described in D2.2.1) and the CARF (described in D2.1.1 and D2.1.2) too.

This meta-model is composed by 32 classes. They are briefly defined as follows:

- **Adapter:** represents the agent or set of agents that is responsible for triggering or deciding the adaptation steps, such as a:
  - **User:** the end user that is interacting with a system in a given moment, a human user;
  - **System:** the computational application that interacts directly with the system;
  - **Third-Party:** an external agent able to intervene in the adaptation process;
- **Context:** all the information that characterizes the context of use, the interaction scenario and that is relevant for the adaptation. It is mainly defined in terms of:
  - **User:** information about the end user, including for instance profile, age, gender, or preferences;
  - **Platform:** the device or set of devices used to interact, and all their characteristics;
  - **Environment:** the scenario in which the interaction is occurring, defined for instance in terms of light level, noise level, etc;
  - **Application:** the system itself, described by models, requirements, task trees, etc;
    - **Element:** one specific object of the context (i.e. a context information);
    - **Property:** one specific attribute that characterizes one element;
    - **Quality:** a metric to evaluate the context information, e.g. validity;
- **Adaptation Process:** the complete set of activities and functions performed to adapt something;
  - **Adaptation Rule:** a formal association between the context and the adaptation techniques;
    - **Justification:** a reason that justifies the context and aids to prioritize the adaptation;
    - **Event:** a specific status or change of status regarding the system or the context;
    - **Condition:** an association between an element and a given instance by means of an operator (e.g. equal, greater than) that enables comparison and evaluation;
    - **Action:** a function that defines the execution of the adaptation;
    - **Technique:** a change in the system or in one or more properties of the system;
  - **Policy:** an abstraction of a technique that governs it;
  - **Strategy:** an abstraction of a policy that governs it;
    - **Operation:** one specific function that changes something (e.g. re-size);
    - **Classifier:** a definition of amount, in terms of percentage for instance (e.g. all, any);
    - **Resource:** a component of the UI or the system that can be subject to adaptation;
      - **Resource Property:** a specific characteristic or attribute of a resource;
    - **Parameter:** a precise value that defines the adaptation technique (e.g. 50%);
  - **Presentation Method:** a explicit manner of presenting the adaptation to the end user;
- **Model:** an abstract representation of the reality (of the system, its different perspectives and the UI):
  - **Task:** a set of actions and activities to be executed in a given order;
  - **AUI:** the abstract definition of the system and its UI;
  - **CUI:** a more concrete definition of the system, its UI and its components;
  - **FUI:** the (graphical) user interface to be presented and/or rendered to the end user.

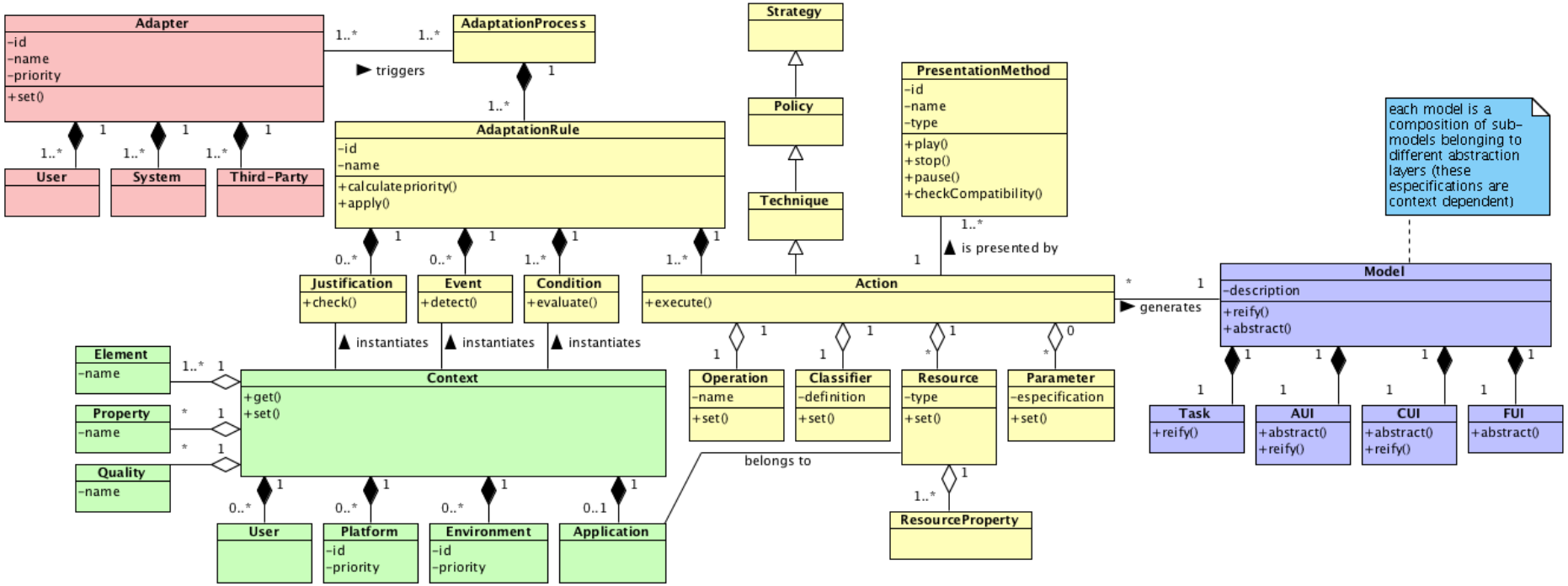

Figure 9. CAMM: a Meta-Model formalizing context-aware adaptation

4.2 Use Case Diagrams

The Use Case Diagrams in the context of Serenoa project aims at three main goals: (i) defining the agents (or roles) that are involved in an adaptation process, (either by triggering or deciding it), (ii) the use cases, or techniques that will be performed by the system regarding the adaptation, and (iii) their possible extensions.

The Use Cases of Serenoa and their respective descriptions were presented in the first release of this deliverable (D3.1.1), in this second release they were updated accommodating some extensions and further techniques for adaptation. The use cases present in this section are associated with the CAMM as an action performed within a given adaptation rule, and with the CARF as an adaptation technique that is listed in the *how* branch. We opted to organize them according to the resource that is subject to the adaptation (i.e. content, navigation and presentation). However, other criteria could also be adopted (e.g. by context information, or software quality of interest).

The Figure 10 illustrates an overview of the use case diagram, in which the agent responsible for the adaptation is abstracted as *adapter*. This agent can also be specialized by specific roles, such as: user, system or a third party. Other specializations, such as developers, or designers can be also considered [Ganneau et al., 2007]. The main use case name *adapt* consists in the changes executed in the application or its UI, including the adaptation of navigation, presentation and contents. The adaptation of the contents is also refined by specific adaptations according to the content format, i.e. audio, image, text, UI element and video.

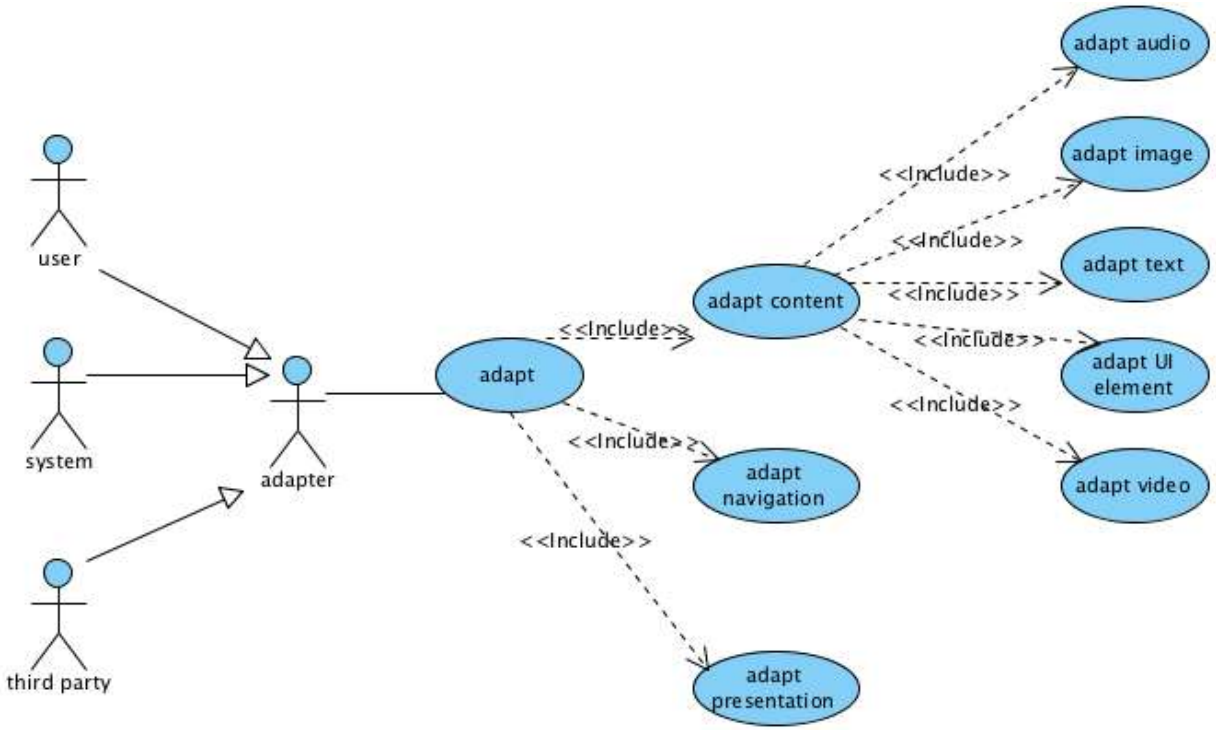

Figure 10. Use Cases Overview: the main agent, named adapter, can be specialized as an end user, a system or a third party. It is responsible for adapting: contents (audio, image, text, UI elements, or video), navigation or presentation.

Each of these adaptations have their specific use cases, they were detailed in further diagrams: Figure 11 illustrates the adaptations for contents in general, Figure 12 for audio contents, Figure 13 for images, Figure 14 for text, Figure 15 for UI elements and Figure 16, for video. The use cases regarding adaptation of navigation and presentations are respectively illustrated by Figure 17 and by Figure 18.

Concerning the adaptation of contents, in general, without any specific format, 12 generic use cases were considered, and 5 possible extensions, refining the techniques of re-sizing and re-ordering according to possible specializations (e.g. re-size the height, or re-order by chronological order). These use cases These use are listed in Figure 11.

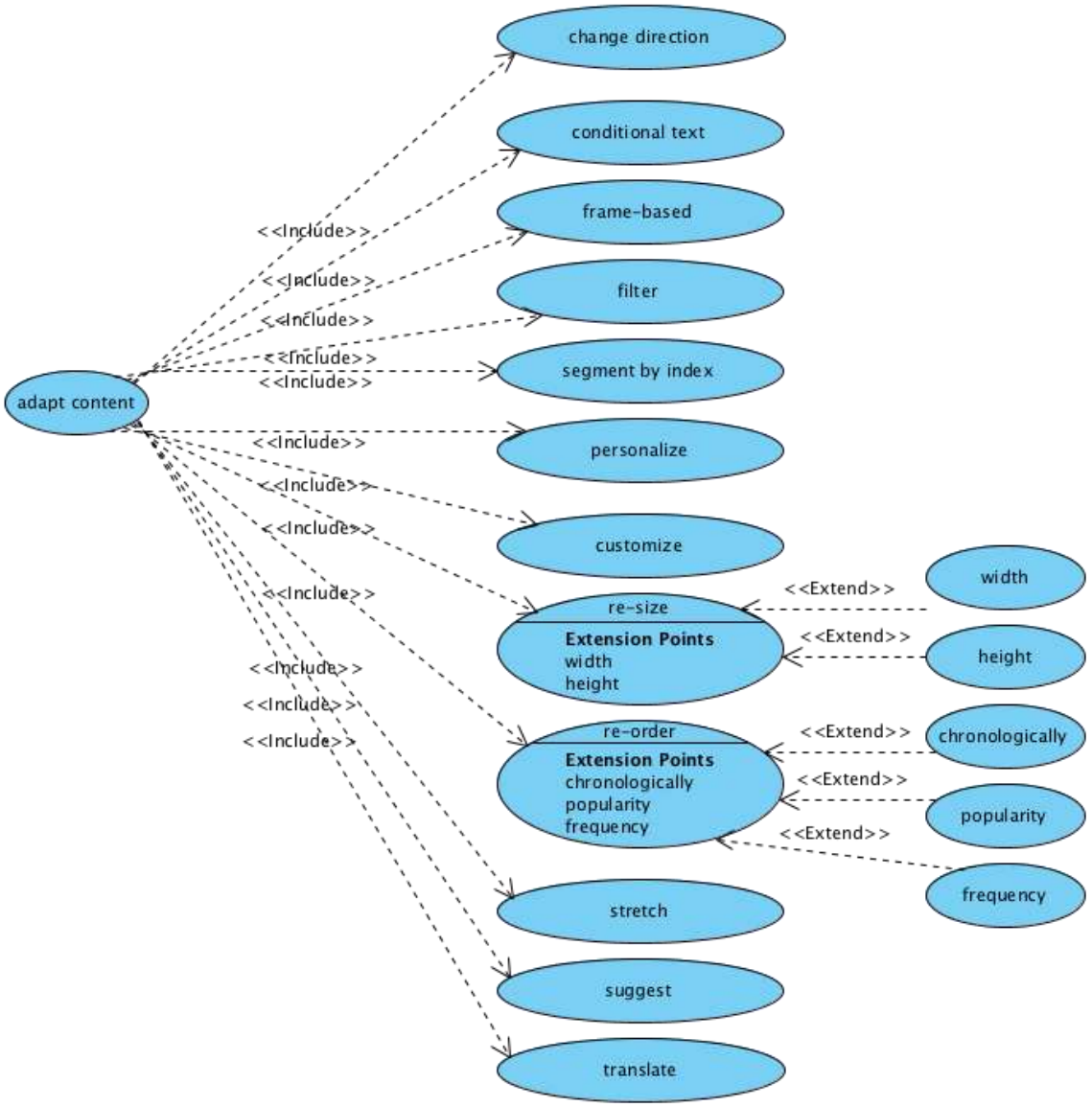

Figure 11. Use Case Diagram for Adapting Content

Concerning the adaptation of audio contents, as Figure 12 illustrates, 7 main use cases were considered and 13 extensions, mainly they consist in changing its properties (such as bit rate, sample rate, volume and speed), converting them (e.g. converting the channel or the format), translating them (e.g. language or modality), replacing (by images, text or video), simplifying, summarizing or truncating (e.g. by time) the contents. Each of these use cases must be applied for a given context of use and according to the capabilities of the device (or set of devices) in use. For instance the volume must be adapted according to the noise levels of the environment in which the user is located, however it can only be replaced by an equivalent image if there is a screen available to present it.

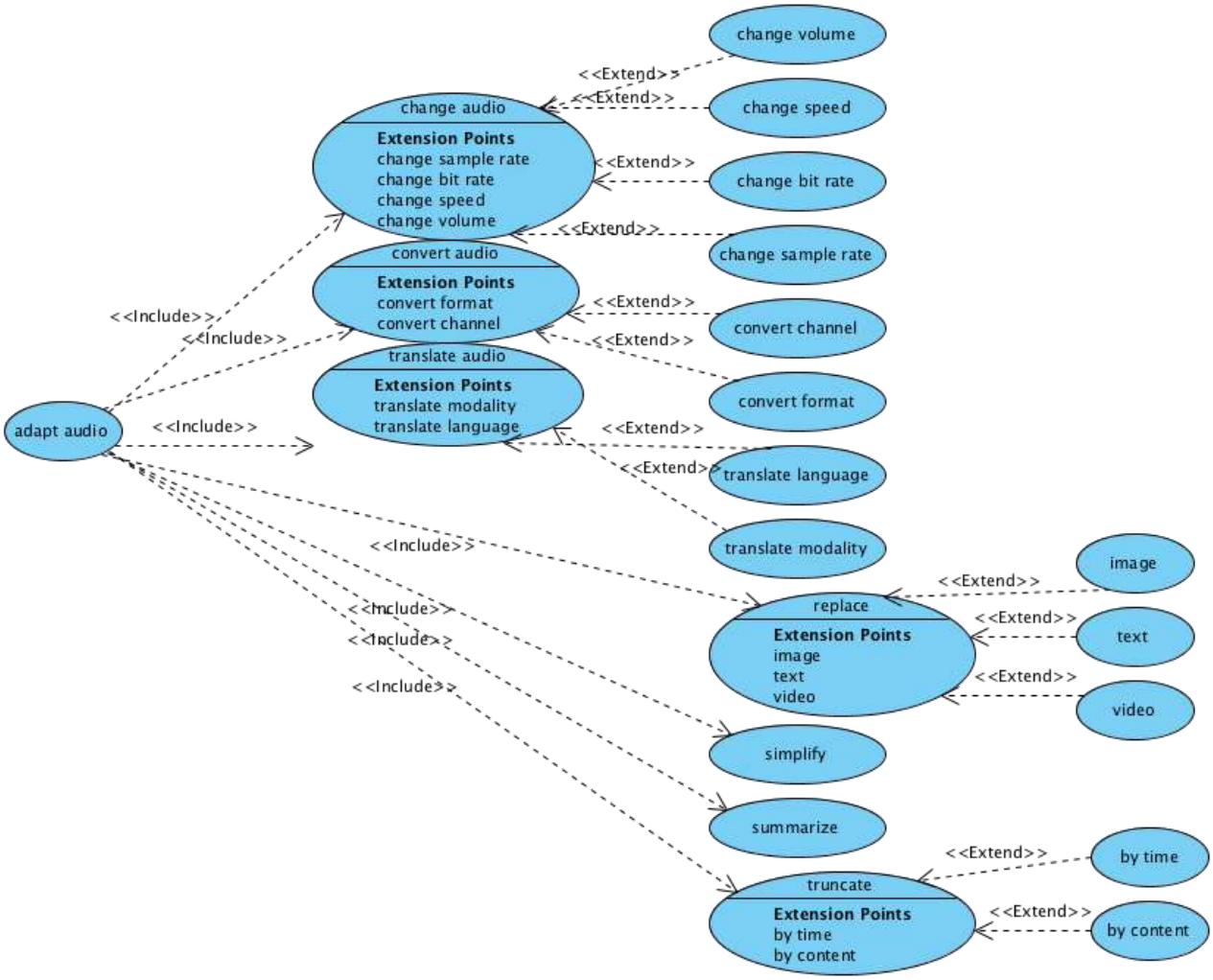

Figure 12. Use Case Diagram for Adapting Audio

The Figure 13 illustrates the Use Case Diagram for adapting images. 21 techniques are included and they are refined in 13 extensions. These use cases involve changes that may affect the complete image, or its specific properties, such as their dimension, or specific colours' properties.

Figure 14 illustrates the Use Case Diagram for adaptation of text. 17 techniques were considered, and 18 extensions. This UCD composes an extensive list of possibilities in terms of context-aware adaptation for textual contents.

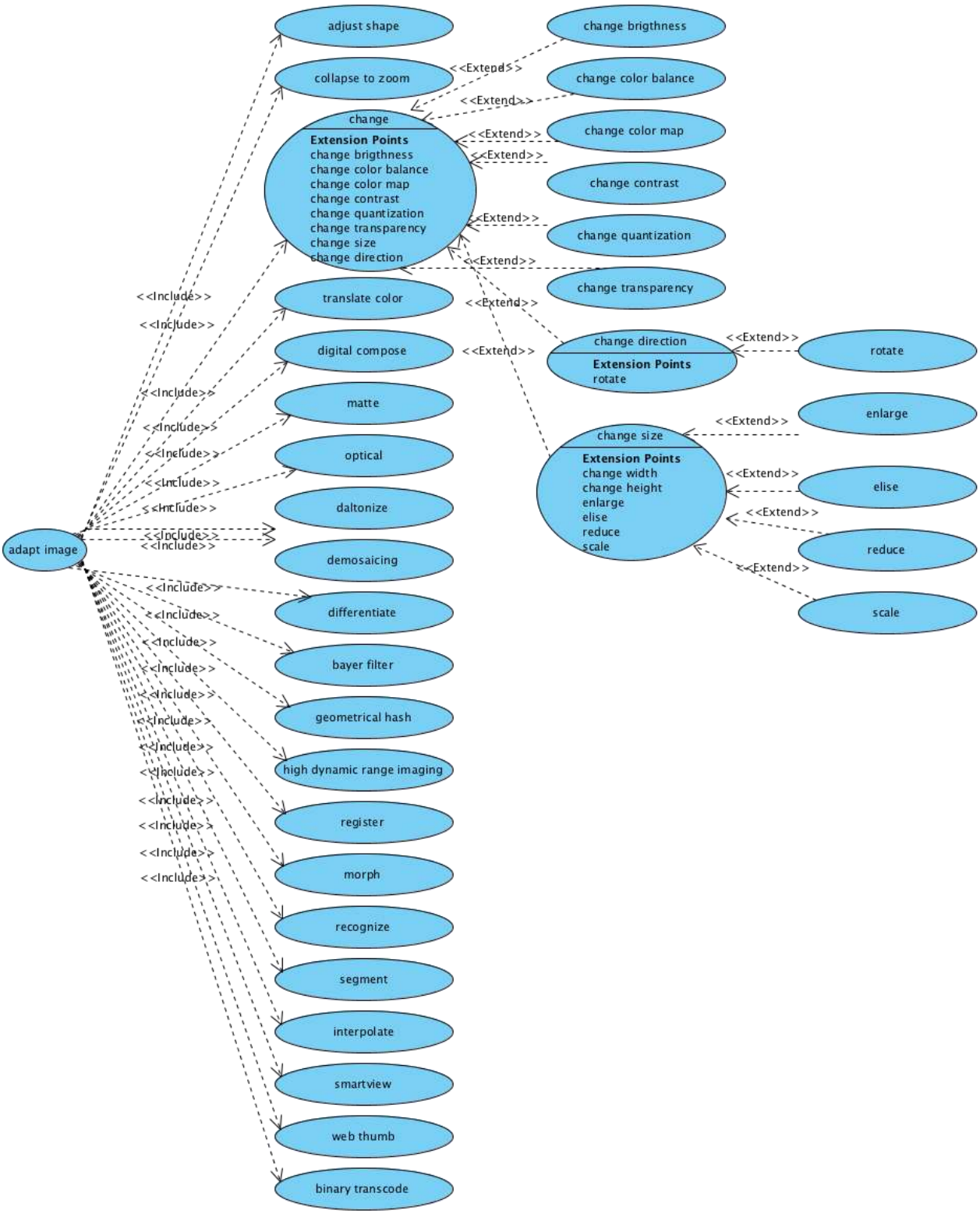

Figure 13. Use Case Diagram for Adapting Image

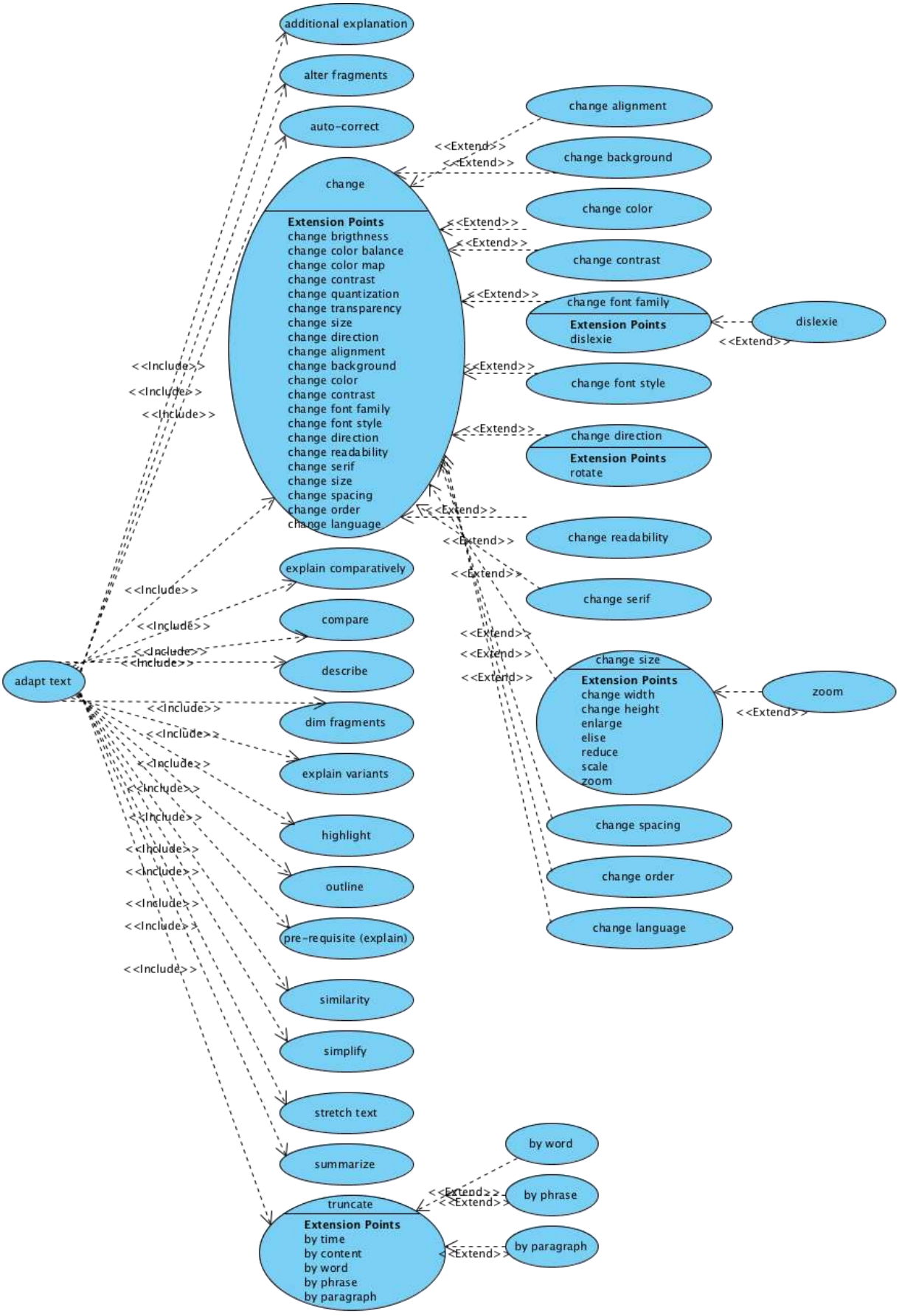

Figure 14. Use Case Diagram for Adapting Text

Figure 15 illustrates the Use Case Diagram for UI elements. 10 techniques are included and 2 possible extensions; they involve different types of widgets, such as forms, text boxes and tables.

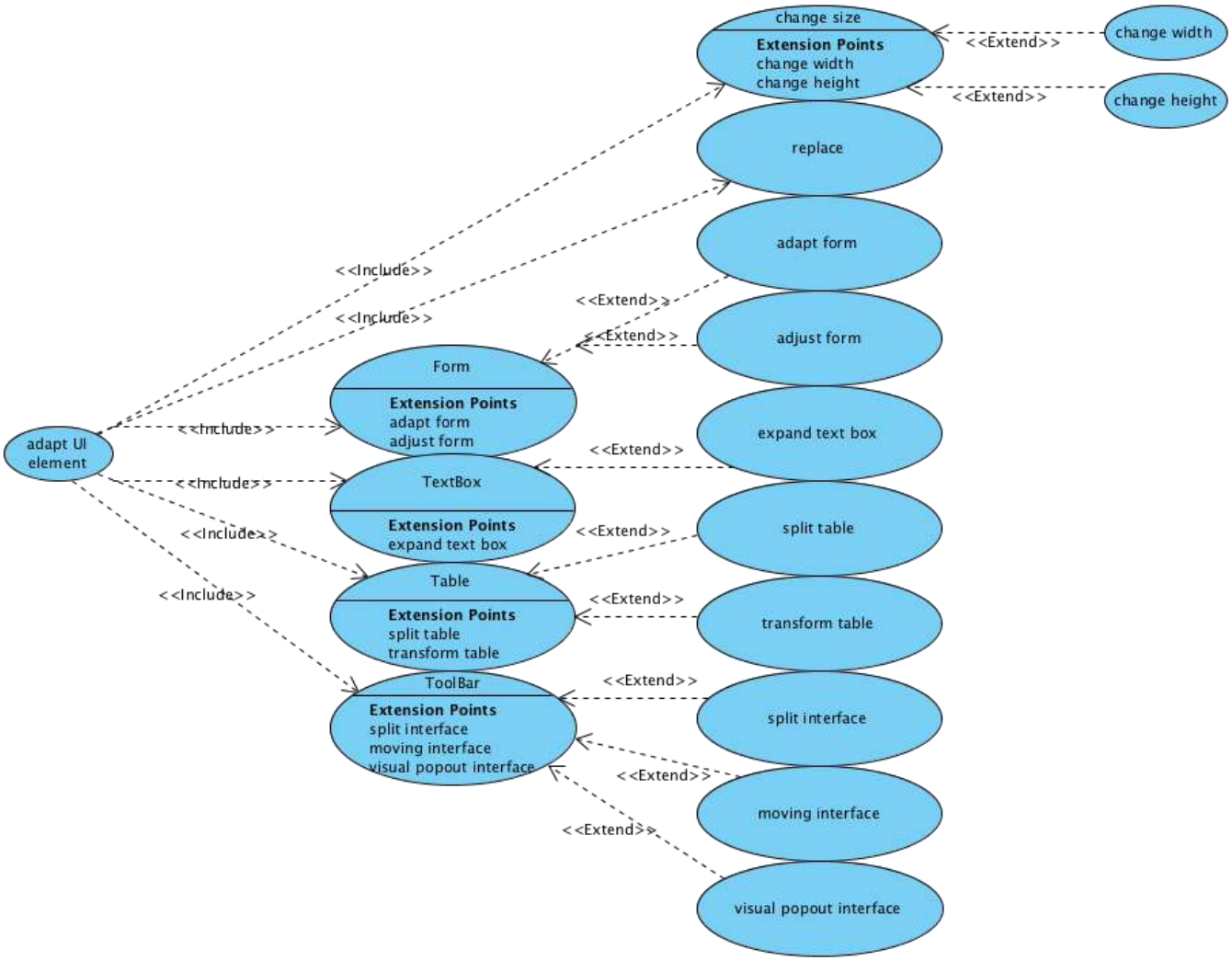

Figure 15. Use Case Diagram for Adapting UI Elements

Figure 16 illustrates the Use Case Diagram for adaptation of video. It includes a set of 7 techniques that are extended by 8 use cases. They affect the complete video (e.g. when it is replaced by an equivalent textual description, ensuring accessibility) or its specific properties (e.g. changes in the video speed, or its frame resolution).

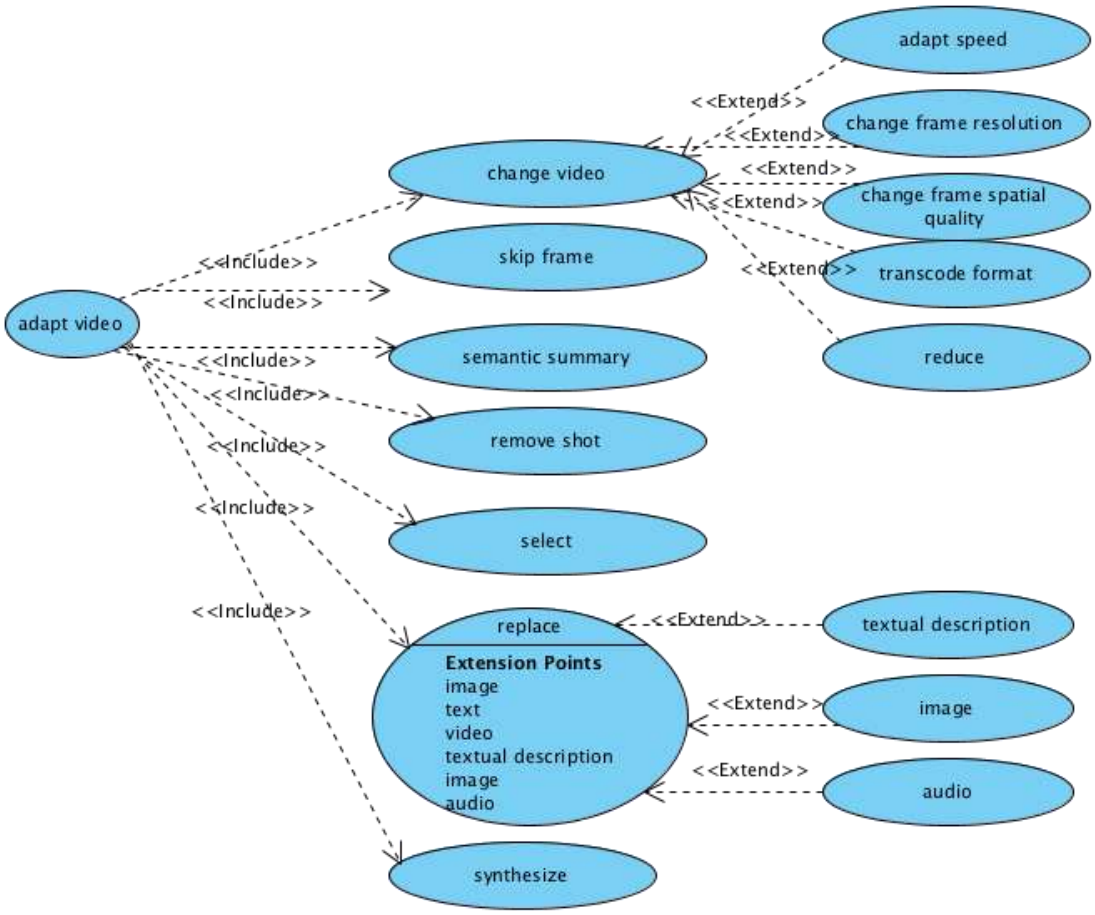

Figure 16. Use Case Diagram for Adapting Video Contents

Figure 17 illustrates the Use Case Diagram for adaptation of Navigation. It includes 19 adaptation techniques and 6 possible extensions.

Figure 18 illustrates the Use Case Diagram for adaptation of Presentation. It is composed by 15 techniques and 8 extensions.

The detailed descriptions of each use case are presented in D3.1.1.

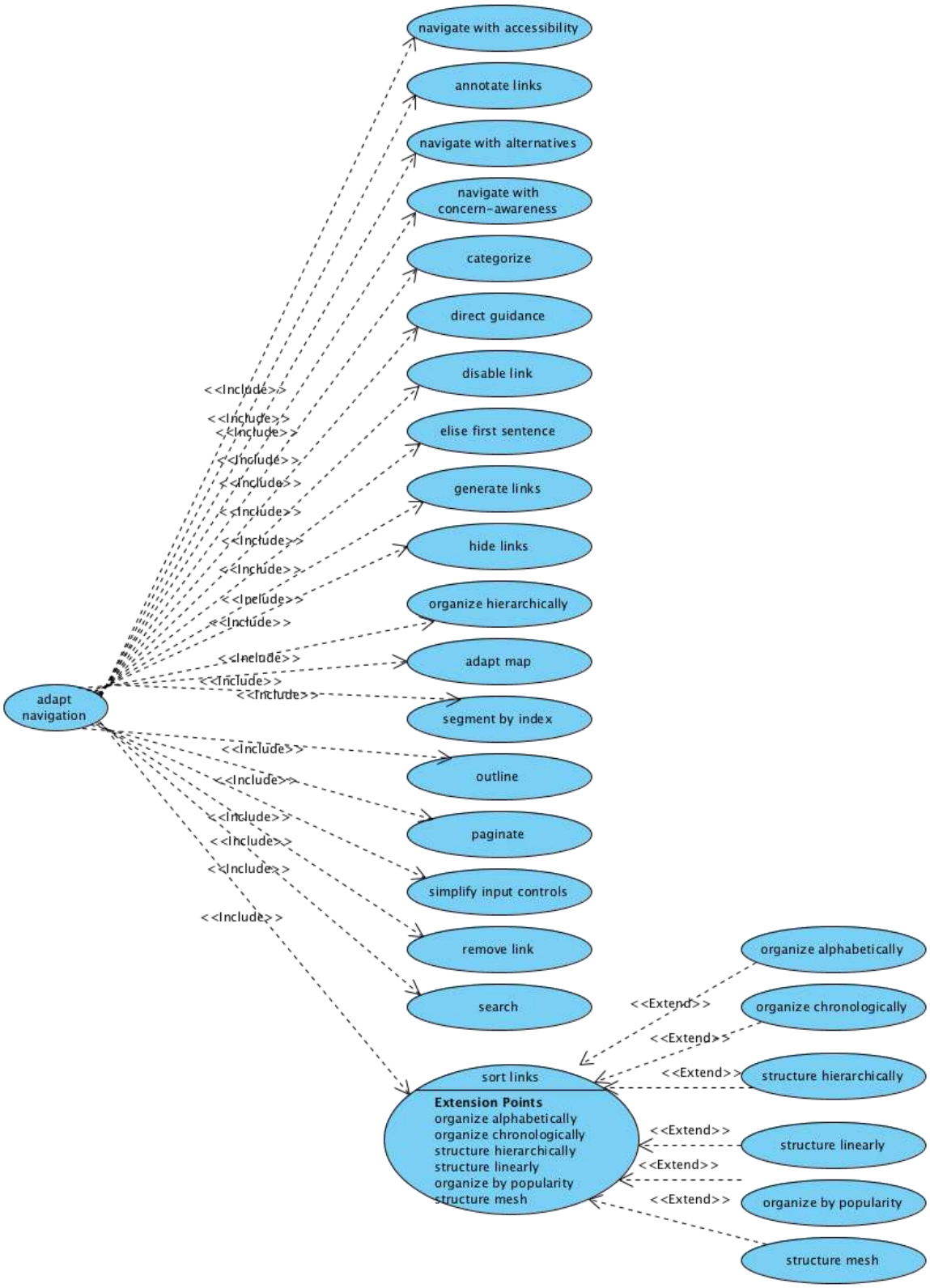

Figure 17. Use Case Diagram for Adapting the Navigation

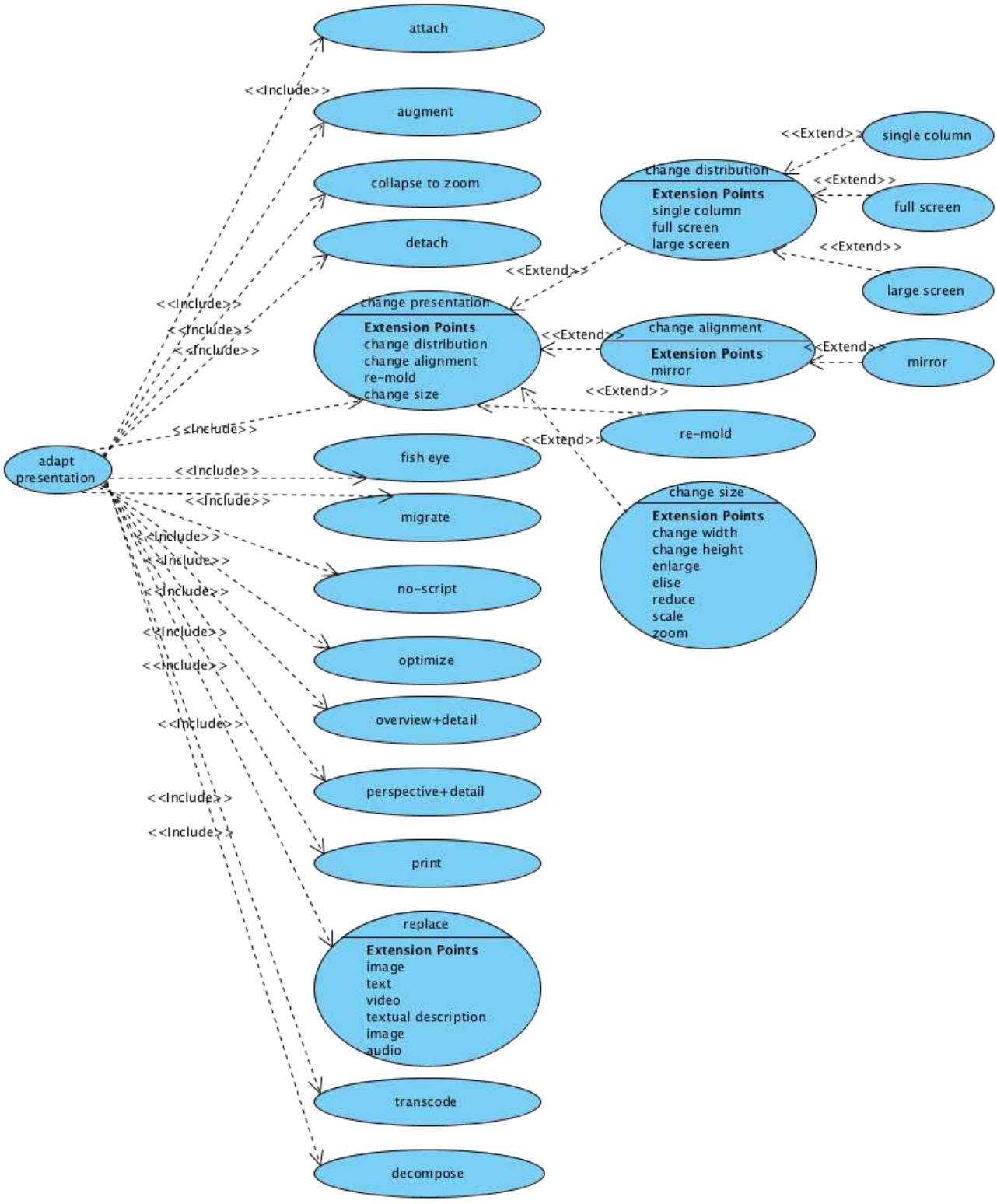

Figure 18. Use Case Diagram for Adapting the Presentation

5 Final Remarks

In this deliverable we briefly presented and discussed a set of related works targeted at defining meta-models for context-aware adaptation or topics that are closely related to it, such as adaptation rules or context-awareness. Based on the analysis of the related works a meta-model for context-aware adaptation was created, maintaining the essential concepts of the previous works, extending and refining their definitions.

This deliverable also presents a new version of the use case diagrams for context-aware adaptation, further techniques and possible extensions were considered in the new release. No updates are reported for the remaining diagrams (i.e. sequence, state-chart and class). They will be presented in the third release of this deliverable (i.e. D3.1.3)

5.1 Conclusion

We believe that by defining a meta-model for CAA it is possible to achieve a consistent and formal abstraction of relevant concepts in a unified view. Such a meta-model can be adopted by stakeholders to facilitate the development and the design decisions for applications that are adaptable or adaptive.

Moreover as emphasized by Luyten et al. (2012), we believe that by using a MOF-compliant meta-model, later integration with other MOF-compliant models becomes also possible.

A combination of graphical diagrams, illustrating the definitions of relevant elements for context-aware adaptation, by means of models and meta-models consists in a powerful manner to describe an adaptation process in a manner that is as specific as possible. We aim at achieving expressiveness and still assuring the readability of the diagrams.

We believe that by adopting a definition of CAA concepts in a high abstraction level, we provide solutions that are both platform and technology independent, covering a general-purpose specification.

5.2 Future Work

As a future work we plan to cross-validate the meta-model by checking its validity in different application domains and with applications of different complexity levels. We also plan to refine such models if necessary.

References

[Booch et al., 1998] Grady Booch, James Rumbaugh, Ivar Jacobson. 1998. Unified Modeling Language User Guide, the (1st Edition) (Addison-Wesley Object Technology Series). Addison-Wesley Professional.

[Calvary, 2003] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User In-terfaces. Interacting with Computers 15, 3 (June 2003), pp. 289-308.

[DCO] José Manuel Cantera Fonseca; Rhys Lewis. Delivery Context Ontology. 16 June 2009. W3C Working Draft. (Work in progress.) URL: http://www.w3.org/TR/2009/WD-dcontology-20090616/

[Dessart, 2011] Dessart, Ch.-E., Motti, V. G., and Vanderdonckt, J. 2011. Showing User Interface Adaptivity by Animated Transitions. In Proc. of 3rd ACM Symposium on En-gineering Interactive Computing Systems. EICS'2011. ACM Press, NY, 95-104.

[Dittrich, 1995] Dittrich, K.R., Gatziu, S., Geppert, A. The Active Database Management System Manifesto: A Rule- base of ADBMS Features. In Proceedings of the 2nd International Workshop on Rules in Database Systems, Vol. 985, Springer-Verlag, 1995, pp. 3-20.

[Duran et al., 2010] J. I. Durán, J. Laitakari, D. Pakkala, and J. Perälä, "A User Meta-model for Context-Aware Recommender Systems," In: Proc. Of HetRec'10, pp. 3–6, 2010.

[Farias et al., 2007] Clever R. G. de Farias, Marcos M. Leite, Camilo Z. Calvi, Rodrigo M. Pessoa, and Jose G. Pereira Filho. 2007. A MOF metamodel for the development of context-aware mobile applications. In Proceedings of the 2007 ACM symposium on Applied computing (SAC '07). ACM, New York, NY, USA, 947-952. DOI=10.1145/1244002.1244209 http://doi.acm.org/10.1145/1244002.1244209

[Fuchs et al., 2005] Florian Fuchs, Iris Hochstatter, Michael Krause, and Michael Berger. 2005. A Metamodel Approach to Context Information. In Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW '05). IEEE Computer Society, Washington, DC, USA, 8-14. DOI=10.1109/PERCOMW.2005.9 http://dx.doi.org/10.1109/PERCOMW.2005.9, Munich, 2004.

[Ganneau et al.] V. Ganneau, G. Calvary, R. Demumieux, 2007. Métamodèle de règles d'adaptation pour la plasticité des interfaces homme-machine. In *Proceedings of the 19th International Conference of the Association Francophone d'Interaction Homme-Machine*(IHM '07). ACM, New York, NY, USA, 91-98. DOI=10.1145/1541436.1541454 http://doi.acm.org/10.1145/1541436.1541454

[Gomez and Tran, 2009] J. M. Gómez and T. Tran, "A Survey on Approaches to Adaptation on the Web," System, pp. 137–153, 2009.

[Horvitz , 1998] Horvitz, E., Breese, J., Heckerman, D., Hovel, D., Rommelse, K. The Lumière Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, July 1998, pp. 256-265.

[Kaklanis et al., 2010] N. Kaklanis, P. Moschonas, K. Moustakas and D. Tzovaras, "Enforcing accessible design of products and services through simulated accessibility evaluation", International Conference on ICT for ageing and eInclusion, CONFIDENCE 2010, Jyvaskyla, Finland, December 2010.

[Lopez-Jaquero et al., 2009] López-Jaquero, V., Montero, F., and Real, F. Designing user interface adaptation rules with T: XML. In Proceedings of IUI. 2009, 383-388.

[Luyten et al., 2010] Luyten, K., Haesen, M., Ostrowski, D., Coninx, K., Degrandsart, S., Demeyer, S.: Storyboard creation as an entry point for model-based interface development with UsiXML. In: UsiXML, pp. 1–8 (2010)

[MORFEO] Available at: http://forge.morfeo-project.org/wiki_en/index.php/Context_Of_Use_Metamodel#Context_Of_Use

[MOF] OMG: Meta Object Facility (MOF) Core Specification, OMG Available Specification, version 2.0. Object Management Group, 2006

[NEXOF-RA] NEXOF-RA Project, http://www.nexof-ra.eu

[Simons, 2007] C. Simons, “CMP : A UML Context Modeling Profile for Mobile Distributed Systems,” Sciences-New York, pp. 1–10, 2007.

[Strang and Linnhoff-Popien, 2004] Strang, T. and Linnhoff-Popien, C. A Context Modeling Survey. In Proc. Workshop on Advanced Context Modelling, Reasoning and Management 2004.

[UsiXML] UsiXML Specification. Available at: usixml.org

[Vanderdonckt et al., 2004] Vanderdonckt, J., Limbourg, Q., Michotte, B., Bouillon, L., Trevisan, D., Florins, M., UsiXML: a User Interface Description Language for Specifying Multimodal User Interfaces, in Proc. of W3C Workshop on Multimodal Interaction WMI'2004 (Sophia Antipolis, 19-20 July 2004).

[Zimmermann et al., 2007] Andreas Zimmermann, Marcus Specht, and Andreas Lorenz. 2005. Personalization and Context Management. User Modeling and User-Adapted Interaction 15, 3-4 (August 2005), 275-302. DOI=10.1007/s11257-005-1092-2 http://dx.doi.org/10.1007/s11257-005-1092-2

Acknowledgements

- TELEFÓNICA INVESTIGACIÓN Y DESARROLLO, http://www.tid.es
- UNIVERSITE CATHOLIQUE DE LOUVAIN, http://www.uclouvain.be
- ISTI, http://giove.isti.cnr.it
- SAP AG, http://www.sap.com
- GEIE ERCIM, http://www.ercim.eu
- W4, http://w4global.com
- FUNDACION CTIC http://www.fundacionctic.org

Glossary

- AKA: also known as
- ArgoUML: tool to support the creation, edition and visualization of UML models
- AUI: Abstract User Interface
- Cameleon RF: A Reference Framework defining UI models of four abstraction levels
- CADS: Context-aware Design Space for Adaptation
- CARF: Context-aware Reference Framework for Adaptation
- CD: Class Diagrams
- CR: cross-reference
- CUI: Concrete User Interface
- FUI: Final User Interface
- GMF: Graphical Model Framework
- MARIA: User Interface Description Languages, XML-based
- MDA: Model Driven Architecture
- MDE: Model Driven Engineering
- MM: Meta-model
- MOF: Meta-Object Facility
- OMG: Object Management Group
- QVT: Query-View-Transformation
- SD: Sequence Diagram
- UCD: Use Case Diagram
- UI: User Interface
- UML: Unified Modelling Language
- UsiXML: User Interface Description Language, XML-based

Further definitions can be retrieved at: http://www.serenoa-fp7.eu/glossary-of-terms/