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

Project no. FP7 – ICT – 258030

Deliverable 3.1.3

Reference Models Specification (R3)

Due date of deliverable: 30/09/2013

Actual submission to EC date: 30/09/2013

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	W3C
Reviewer 1	
Reviewer 2	
Approved by	
Project Officer	Michel Lacroix

Contributors	
Partner	**Contributors**
UCL	Vivian Genaro Motti, Nesrine Mezhoudi, Jean Vanderdonckt
W3C	Dave Raggett

Changes			
Version	**Date**	**Author**	**Comments**
1	30/08/2013	UCL	Basic structure of the document: table of content, initial content based in the Description of work
2	04/09/2013	UCL	Completed version of the diagrams, synchronization of the descriptions
3	12/09/2013	W3C	Review of the contents and structure
4	13/09/2013	UCL	Final Reviewed Version

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. The third and final release of this deliverable consolidates the models, presenting a more complete and refined version of all diagrams previously presented.

Table of Contents

1 Introduction

1.1 Objectives

This deliverable is dedicated to present the consolidated version of all reference models for context-aware adaptation (CAA) that has been previously reported. Such models formalize fundamental concepts, which define the implementation and the execution of 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. Finally, this third release consolidates the previous model, by presenting a more complete and refined version of the reference models.

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 3.1.2 – Reference Models (R2) presents the second 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 Introduction

Users interact from different environments, using different platforms and devices, and have also different profiles. This multitude of scenarios in which the interaction takes place poses a challenge not only for developers but also for end users. Developers are not always aware about how to handle such differences properly, then they do not know how to prioritize the possible contexts, moreover implementing one single version of each system UI for each specific context is nor feasible neither scalable. However if such constraints and characteristics are ignored while implementing applications, it is likely that the end users access will be reduced, hindered or even prevented. In order to support stakeholders in the development of UI's that are able to adapt themselves properly, we propose the adoption of a unified model, i.e. by starting from a formal abstraction of interactive systems that support adaptation, the navigation, presentation and contents of system can be adapted. A model-based approach enables the generation of different versions of the same application, in an automatic or semi-automatic manner, properly accommodating the different requirements imposed by different interaction scenarios. For instance the same application for renting a car can be used in a smartphone, a tablet PC or a Desktop, by a young or an elderly user. The work presented in this paper relies on the assumption that in spite of the requirements required in different contexts vary, a model-based approach can efficiently leverage the efforts needed to generate adapted versions of a system.

2.2 Context-aware Adaptation

Context-aware Adaptation (CAA) involves the identification of the relevant context information that surrounds the user during her interaction in order to properly adapt elements of an interactive system aiming at enhancing the end user interaction. The main goals of CAA are improving the usability levels of the system by using the relevant information of the user context to properly transform a system.

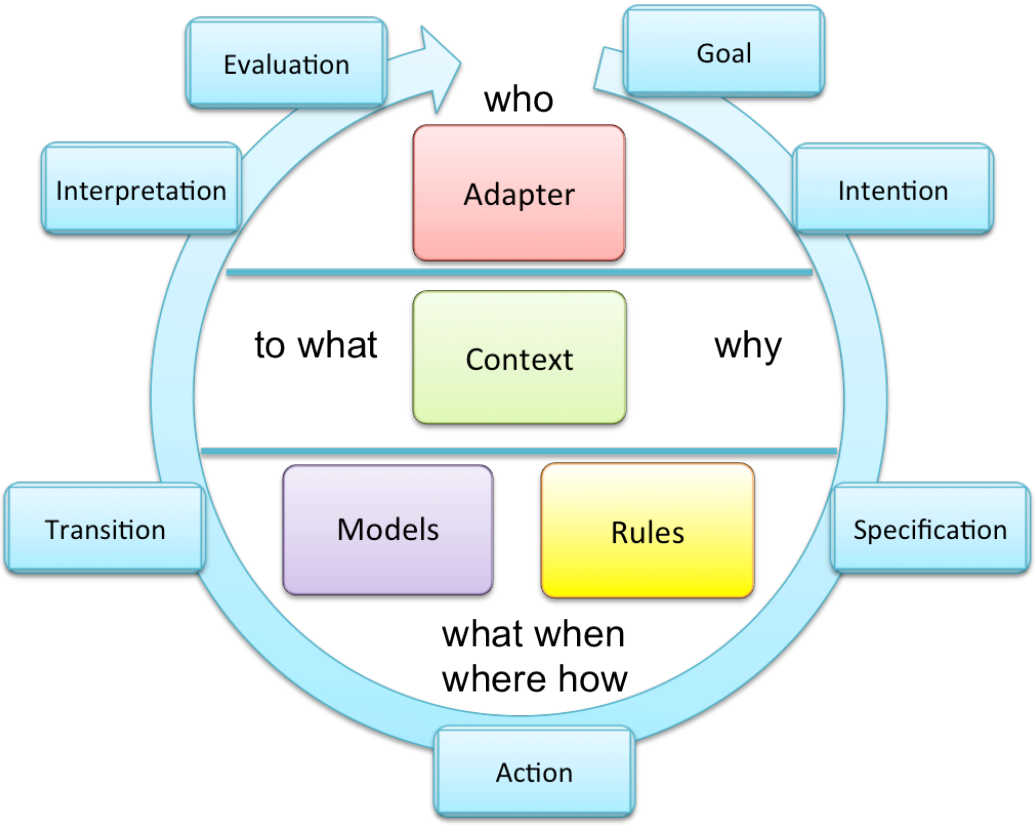

Figure 1. Adaptation lifecycle structured according to: its four core components (Adapter, Context, Models and Rules), seven reference information (who, to what, why, what, when, where, and how) [Mot13] and seven phases (goal, intention, specification, action, transition, interpretation and evaluation) according to Norman's theory of action [Nor86]

Figure 1 illustrates the 7 phases of an adaptation lifecycle (according to Norman's theory of action [Nor86]) relating it to its 4 core components (adapter, context, models and rules) and respective reference information (who, to what, why, what, when, where and how) [Mot13]. On the top of the cycle the adapter (who), i.e. the agent responsible for triggering or deciding the adaptation, has a goal and an intention in mind. Based on this information (why) and in what (to what) context information that surrounds the interaction moment, the system, by means of rules, can specify the actions that will change (how) one or more elements (what) of the interactive system, at design or run time (when), at the client, server or proxy (where). A transition can be then used to present the results of the adaptation in the models to the end user. Such results will be finally interpreted and evaluated by the agent (i.e. the end user, the system, or a third party). Depending on the evaluation given, the adaptation cycle can be concluded (if satisfactory results have been obtained), or continue (until satisfactory results are reached).

Although CAA aims at better usability levels and at improving the user experience, many drawbacks can be foreseen, e.g. (i) there are sets of contextual information that surround the user and it is hard to select or prioritize the most relevant ones, (ii) the processing of the adaptation may have a cost, so the benefits of its results must be greater to compensate for it, (iii) the UI and system changes, however the user should not feel confused, lost or out of control during her interaction, (iv) CAA involves several variables it is up to the designer to define the adaptation with an optimal (and not maximum) amount of information available about the user context, (v) the contexts evolve dynamically, posing a continuous challenge for developers to keep themselves updated regarding novel technologies that may rise and potential approaches to handle them. Furthermore, there is no single approach or ready-to-use solution about CAA, so developers must dedicate a significant effort to search for and retrieve important information about CAA.

2.3 Modelling

2.3.1 Definition and Goals

Models aim at simplifying, abstracting and formalizing systems concepts, including their properties, methods, relationships, cardinalities and constraints. A model is a simplification of reality, i.e. a semantically close abstraction of a system, the objective being to better understand the system being created [Koc00]. On one hand they provide a version of the system to be that is sufficiently complete to comprehend the system goals, on the other hand by being a simplified version of the reality, system models may lack important details about the concepts of interest. Humans adopt models to better deal with complex information and processes, and also to simplify the reality in a way that it can be better handled or processed. We propose a unified model for CAA because so far there is no work that covers at once all its essential concepts and its complete lifecycle. A meta-model can serve as a basis for developers to instantiate their own applications following a consistent structure. The resulting applications may be also more compatible, extensible and flexible due to the fact that different applications will comply with consistent definitions. Besides this, a meta-model enables a (semi)automatic generation of an application, reducing development efforts, time to market, inconsistent results, and facilitating the reuse. The next section summarizes the state-of-the-art of CAA models.

2.3.2 Related Works: Models and Meta-Models

Due to the wide range of application domains, aspects and contexts of use, it is not scalable for the human programmers to create UI versions for each CAA scenario. Instead, an automated solution is necessary [Gaj04]. In this sense, different models have already been proposed to facilitate the design, implementation, execution and evaluation of CAA. This section exemplifies models that support CAA, and that inspired the design decisions and requirements for creating CAMM.

Models abstract system concepts and their relationships. In the domain of CAA, models have been used to represent: context information, adaptation rules, and multimodal properties. This section briefly summarizes, in a chronological order, a selection of seven models that support specific concepts of adaptation.

- **Munich.** a reference model defines techniques for designing adaptive hypermedia applications. The domain model requires a conceptual design of the problem domain, which evolves into a navigation and presentation model. The user model defines attributes and relationships with the domain model. The adaptation model specifies domain and user elements, the set of acquiring and adaptation rules and their collaborations [Koc00].
- **Customization Model.** Kappel et al. (2002) model the customization according to context regarding user profile, network and location. For them context provides relevant information about an interactive application and the environment. Context influences requirements elicitation and triggers customization according to context changes.
- **ADAPTS.** explicitly models task, domain and users, in an integrated manner aiming to support the adaptation according to the context. A diagnostic engine employs the user and expert models to update the navigation selecting the most appropriate tasks for the user based on pre-defined weights [Bru02].
- **Adaptation Model.** Vrieze et al. (2004) base on dynamic behaviors of the user to handling events and use ECA rules to pull and push adaptations in a more flexible fashion. It focuses on hypermedia systems.
- **Context Information.** Fuchs et al. (2005) created a meta-model that defines context information and its associations. The main concepts considered include: devices and persons, their properties, e.g.: mobile phone, phone number, gender, and their relationships, e.g.: is located nearby, has phone number, has last name or is supervised by.
- **Comets.** Calvary et al. (2005) state that an adaptation model specifies evolution and transition rules to be applied if the context changes. They propose adaptation models for defining tasks, abstract, concrete and final UIs and widgets extensions, always considering plasticity as the main principle. They remark the benefits of using model-based approaches to implement CAA, and they also emphasize the adoption of certain principles, namely: plasticity and continuity.
- **CAWAR.** Fahrmair et al. (2005) proposed a calibrateable context adaptation model for ubiquitous applications. It includes the context sensors, interpreters and also actuators.
- **User Model.** Kobsa (2007) identified required characteristics for a user model for adaptation. They include domain independence, inference and reasoning capabilities, support for quick adaptation, extensibility, and privacy support. As future trends for this domain they remark ubiquitous and mobile computing, and smart appliances.
- **Mobile Applications.** Farias et al. (2007) defined a MOF -model for context-aware mobile applications. The concepts considered are abstract and include: classifier, attribute, entity, contents, associations, dependencies, groups and constraints.
- **Adaptation Rules.** Ganneau et al. (2007) and Sottet et al. (2007) created a meta-model that defines adaptation rules, targeting at plasticity as a goal for ubiquitous applications. This meta-model helps designers to take decisions and to implement CAA considering three phases: the context perception, the reaction, and the learning. The rules respect the ECA structure (i.e., on event if condition do action). After the adaptation is defined, the users are able to request, accept or reject it.
- **Adaptation Rules.** López-Jaquero et al. (2008) represented adaptation rules by means of a meta-model that includes concepts as: preconditions, events, sensors, data, transformation and transformation rules.
- **UsiXML.** supports an MDE approach to cover all models required for user interface analysis and design, targeting the context of use, a dynamic entity, whose models are usually subject to continuous changes [Luy10]. 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) [Lim05].
- **Context of Use.** A generic model for context was created for Morfeo project. This model represents elements, properties, entities, aspects, components, characteristics, descriptions of environment and user [Mor12].

The models briefly presented in this session were selected based on their similarity with the topic of interest for this work, i.e. modeling CAA. Once they target at specific concepts of CAA, they complement or specialize each other in a certain way. For instance, the meta-model of Farias et al. (2007) can be seen as a

specialization of the work of Fuchs et al. (2005), Calvary et al. (2005) and [MOR12]. Although the works of Ganneau et al. (2007), Sottet (2007) and López-Jaquero et al. (2008) all focuses on CAA rules, the formers are more specific, respectively targeting at the adoption of principles and at the user feedback.

Table 1 summarizes the works presented above and highlights their focus. They can be broadly organized in two groups: while [Vri04, Gan07 and Lóp08] focus on rules, [Kap02, Cal05, Fuc05, Fah05 and Mor12] focus on context. More specifically [Koc00, Bru02 and Kob07] focus on user models, [Lim05] at platform models and [Far07] targets at mobile devices. Such works are relevant to define essential concepts for adaptation; however by being specialized they provide a narrowed view of the adaptation lifecycle, i.e. by focusing in one specific part of the process, a global definition is still missing.

Table 1. Meta-models for CAA and their main focus

Meta-Model	Focus
Munich Reference Model [Koc00]	User models (preferences, tasks, goals, experience) for adaptation, includes also rules.
Customization [Kap02]	User profile, network and locations.
ADAPTS [Bru02]	Task, domain and users. Tasks are then selected based on the user model.
Adaptation Model [Vri04]	Focuses on hypermedia systems, considers user models and ECA rules to pull and push adaptations.
Context Information [Fuc05]	Defines rules, context in terms of device and person, quality and associations.
CAWAR [Fah05]	Models the adaptation in terms of context interpreters, sensors and actuators, focusing on ubiquitous computing.
Comets [Cal05]	Context-aware adaptation and MBUI oriented to the plasticity of interactive systems.
Generic User Models [Kob07]	User modelling systems characteristics of architectures, requirements and trends.
Mobile [Far07]	Context-aware mobile applications.
Rules [Sot07, Gan07]	Evolution and transition rules based on context's changes. Adaptation rules for ubiquitous computing.
Rules [Lop08]	Rules in terms of conditions and transformations.
UsiXML [Lim05, Luy10]	Considers the context as a dynamic entity and the platform characteristics mainly.
MORFEO [Mor12]	Context-awareness and the users' profile.

Table 1 presents an overview of the main targets, goals and focus of the related works analyzed. To identify which are the relevant concepts to compose the common ground for generating the CAMM (context-aware meta-model) the works reported here were analyzed in depth and the results obtained with such an analysis are presented in Table 2.

Table 2 lists which are the concepts concerning specially: (i) Adapters (system, third-party, user), (ii) Context (user, platform, environment, application domain), (iii) Rules (justification, event, condition, action), and (iv) Models (task and domain, abstract, concrete, final) that have already been explicitly covered by previous works while modeling context-aware adaptation. The works on ADAPTS [Bru02], Generic Adaptivity Model [Vri04], and Generic User Models [Kob07] have not been included in the Table 2 since they do not provide a meta-model of the process itself [Kob07] but task, domain and user models [Bru02], or an overview of the architectural structure [Vri04].

The check signs (✔) indicate when the concept has been explicitly presented as a class in the respective model, and the gray background indicates concepts that are expressed by a generic class, instead of

representing all internal components.

By analyzing Table 2 we notice that most of the works targeted at modeling context-aware adaptation covers mainly context and rules, however the adapters, i.e. who is responsible for the adaptation is often omitted and the resulting models of the adaptation are also often neglected.

Table 2. Comparison of meta-models for adaptation according to their concepts covered specifically (marked with check signs '✔) or generically (grey background). The headers are composed as follows: Adapter (System, Third-Party and User), Context (User, Platform, Environment and Application Domain), Rules (Justification, Event, Condition, and Action), and Models (Task – Domain, Abstract, Concrete, Final)

	Adapter			Context				Rules				Models			
	S	TP	U	U	P	E	AD	J	E	C	A	T D	A	C	F
Koch, 2000			✔	✔						✔	✔	✔	✔		
Kappel, 2002				✔	✔	✔			✔		✔				
Fuchs, 2005				✔	✔	✔									
Fahrair, 2005											✔				
Calvary, 2005				✔	✔	✔						✔	✔	✔	✔
Sottet, 2007									✔	✔	✔				
Farias, 2007															
Lopez, 2008									✔	✔					
MORFEU, 2010				✔	✔	✔									
UsiXML, 2010				✔	✔										

To cover the adaptation lifecycle in a more complete fashion, we propose a Context-aware Meta-model, named CAMM, which explicitly includes all the concepts introduced in the previous sections. The description of CAMM and its internal components are presented in Section 4.

3 Context-aware Meta-model for CAA (CAMM)

The CAMM meta-model has been developed in an attempt to cover the complete adaptation lifecycle, starting from gathering context information and finishing with the generation of the user interfaces (UI's) in a model-based approach. The end user is considered in this cycle, implicitly or explicitly in charge of adaptation decisions. Four main parts compose the meta-model diagram: the context, the agents, the adaptation process and the generation of the UI's.

The concepts defined were selected based on the review of the related literature and by checking whether different instantiations could be supported and expressed by this model, independently of their domains and contexts. This meta-model is composed by 4 main concepts that can be organized as packages. The Adapter i.e. the agent responsible for the adaptation process. The context i.e. all relevant information for adapting the system. The rules that associate the contextual information with the techniques for adaptation. And the models that are subject to the changes applied by the rules. Figure 1 illustrates the main relationships between such concepts.

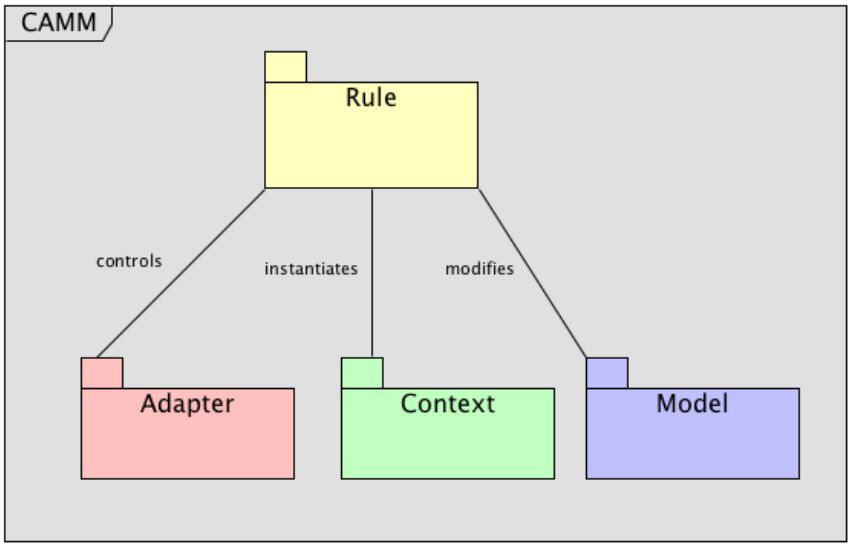

Figure 2. CAMM Packages: Adapters controls the Rules, that are instantiated by the Context and when applied modify the Models

3.1 Overview

Aiming to formalize the most essential concepts that are necessary to implement and execute CAA, a meta-model was created; Figure 3 illustrates it. This meta-model, named CAMM, uses the OMG notation for UML Class diagrams: MetaObject Facility (MOF). In this notation the associations are 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; it abstracts the necessary concepts, establishes their relationships and defines their properties. Besides this, additional information, such as constraints and cardinality of the relations, are also specified.

The CAMM was created based on the analysis of the systematic review results. Four colors were applied in the 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 classes refer to the context of use, the yellow classes refer to the core of the adaptation process, and purple classes refer to the model generation.

This MOF-based meta model diagram, as Figure 3 illustrates, shows with the red blocks three possible agents to trigger an adaptation process: the system, the end user or a third party. These agents are abstracted as 'Adapter'. Considering that an adaptation process may be composed by several phases, different agents can be responsible for each of them [Hor99]. 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. When a group of users is responsible for the adaptation, a crowd-sourced adaptation takes place [Neb11].

A CAA process can also be triggered by a change in the context of use. The green classes 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).

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 classes, can be syntactically structured in the form of ECA rules (event, condition and actions) [Dit95], instantiated and triggered by context information. Event-Condition-Action (ECA) is commonly used and adopted in several workflow prototypes as a modeling tool. The limitation of capturing rules evaluation context in ECA rules leads to the usage of JECA rules where justification (J) provides a reasoning context for evaluations of ECA rules in order to support context dependent reasoning processes in dealing with uncertainties [Nge07]. While conditions and actions are mandatory in a rule definition, justification and events are optional, complementing the rules with additional information. Due to the fact that more than one rule is normally applied simultaneously, conflicts may appear. In order to solve them, one possible approach consists in assigning priorities to the rules based on the related context, another approach is abstracting adaptation techniques in policies (i.e. with 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.

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 [Des11].

As consequences of the actions performed by rules, models for SFE are generated. In the meta-model diagram, the models are represented by purple classes. 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 [Cal03]. 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 colors.

CAMM is composed by 34 classes organized in 4 main packages, 3 enumerations, which are defined in Figure 2.

3.1.1 Adapters

The Adapters, represented by red classes in Figures 1 and 2, refers to the agent or the set of agents that is responsible for triggering or supporting the decisions for the adaptation phases, they are defined as follows:.

Adapter

- **Definition:** the agent or the set of agents that is responsible for triggering or supporting the decisions for each of the adaptation phases;
- **Examples:** the end user that customize the interactive system;
- **Attributes:** id (the identifier of the adapter, a unique value), name (the name associated to the adapter), and priority (could be in a qualitative approach a value
- **Methods:** get() and set() (generic functions used to retrieve the information about the adapters available in a given moment and to associate it to the attribute values, instantiating the adapter);
- **Relationships:** is_composed_by one or a set of *User*, *System*, and *Third-Party*, and triggers an *AdaptationProcess*

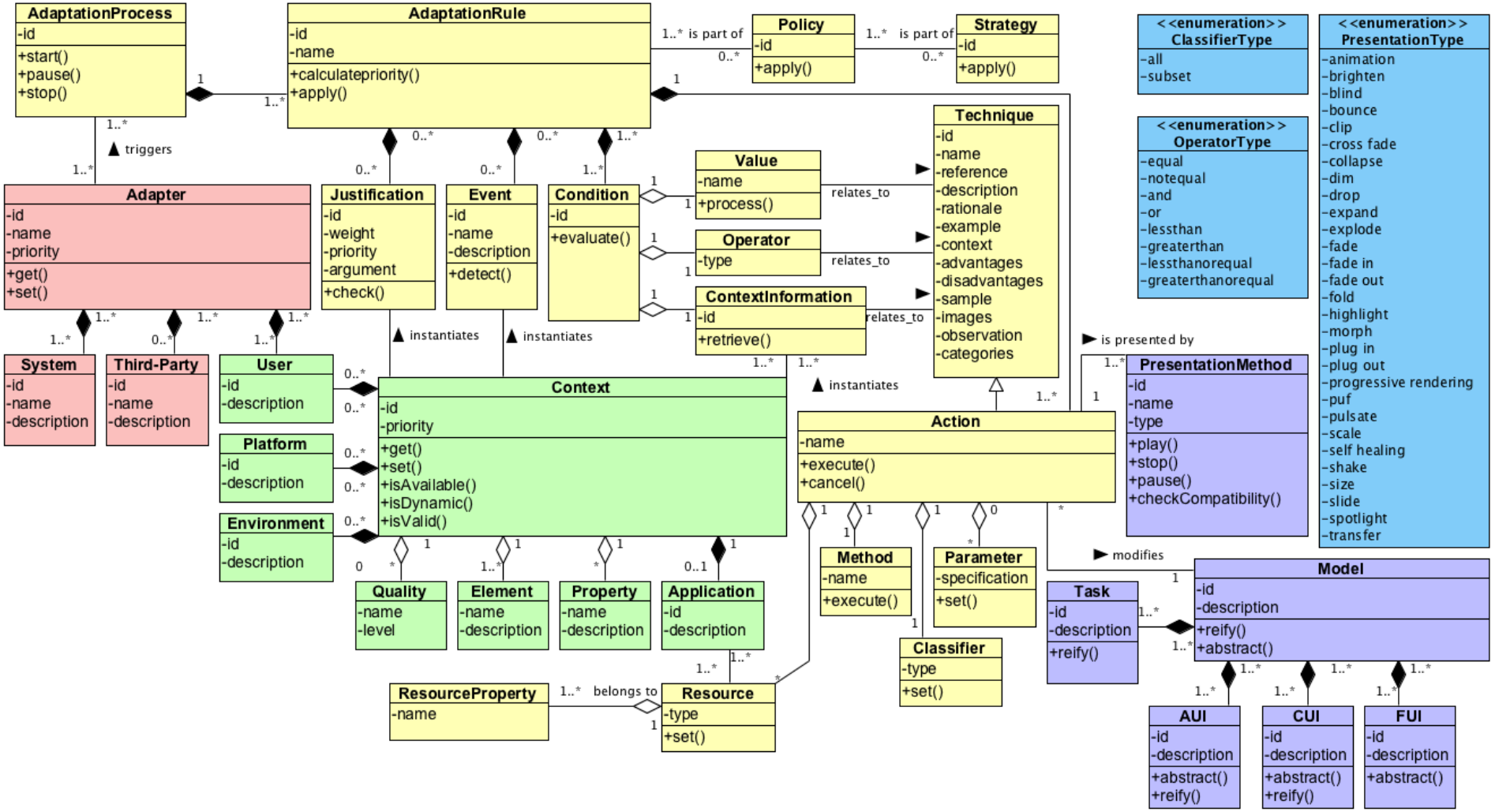

Figure 3. Context-aware Adaptation Meta-model (CAMM)

System

- **Definition:** the computational application (e.g. a function, a program or an API) that interacts directly with the system;
- **Examples:** a web service;
- **Attributes:** id (the identifier of the system, a unique value), name (the name associated to the system) and its description (a summary of its definition and goals);
- **Methods:**
- **Relationships:** composes one or a set of *Adapter*

Third-Party

- **Definition:** an external agent able to intervene in the adaptation process;
- **Examples:** an agent;
- **Attributes:** id (the identifier of the third-party, a unique value), name (the name associated to the third-party) and its description (a summary of its definition and goals);
- **Methods:**
- **Relationships:** composes one or a set of *Adapter*

User

- **Definition:** the end user that is interacting with a system in a given moment, a human user;
- **Examples:** John Doe is the end user interacting with the system, his description includes his personal information, impairments (cognitive, motor, visual, etc.), and preferences;
- **Attributes:** id (the identifier of the user, a unique value) and its description (a summary of its definition and goals);
- **Methods:**
- **Relationships:** can compose one or a set of *Adapter* and also one or a set of *Context*

When several users are considered in the decision, the adaptation is classified as crowd sourced [Neb11], and a mixed approach occurs when a combination of agents collaborate to take the adaptation decisions [Hor99]. The user is part of both *Adapter* and *Context*, first as the agent responsible for taking adaptation decision and then its description is also relevant to composed contextual information.

3.1.2 Context

The context, represented by green classes in Figure 2 and **Error! Reference source not found.**, refers to all the information that characterizes the context of use, the interaction scenario and that can be relevant for defining and executing the adaptation. It is defined mainly in terms of:

Context

- **Definition:** all the information that characterizes the context of use, the interaction scenario and that can be relevant for defining the adaptation lifecycle;
- **Examples:** the user John Doe interacting with a tablet PC in a train;
- **Attributes:** id (a unique identifier value for that context), and a priority value (if in a qualitative approach could be high, medium, low levels, this information is useful to solve potential conflicts between adaptation techniques)
- **Methods:** get() (to retrieve information about the context, coming for instance from sensors in the environment), set() (function to instantiate the values for the context, such values can be treated

and processed beforehand if necessary, e.g. to convert units), isAvailable() (to check whether there is information to be retrieved), isDynamic() (to check whether the information varies along the time), isValid() (to check whether the information is still holding);

- **Relationships:** is_composed_by at least one but not necessarily all *User*, *Platform*, *Environment* and *Application*, aggregates a *Quality*, an *Element*, a *Property*, and instantiates a *Justification*, an *Event* and a *ContextInformation*

User (see previous definition at Section 4.1 for information)

Platform

- **Definition:** the device or set of devices used to interact, and all their relevant characteristics as accessories available, connections, technologies supported;
- **Examples:** a tablet PC with Android, specification about the connections, ports, compatibility, drivers, etc.;
- **Attributes:** id (unique identifier associated to the platform) and its description (a brief summary of the devices available and their characteristics);
- **Methods:**
- **Relationships:** a set of *Platform* can compose a set of *Contexts*;

Environment

- **Definition:** the scenario in which the interaction takes place, defined for instance in terms of light level, noise level, stability level, location, etc;
- **Examples:** a train, with medium noise, light and stability level;
- **Attributes:** id (unique identifier associated to a given environment) and a description (a brief summary of the situation concerned and its characteristics);
- **Methods:**
- **Relationships:** composes *Context*;

Application

- **Definition:** the description of the interactive system and its domain, described by (domain and data) models, (functional and non-functional) requirements, task tree, etc;
- **Examples:** a safety-critical system, a medical system;
- **Attributes:** id (unique identifier associated to the environment) and its description (a brief summary of the characteristics of the environment);
- **Methods:** is_contained_by *Context* and is_associated_with *Resources* (the components of the UI's of the interactive system)
- **Relationships:** composes *Context* and has *Resources*;

Quality

- **Definition:** a qualitative value used to evaluate certain characteristics of the context information (e.g. its validity, availability, precision);
- **Examples:** the information has a high level of precision (adopting a qualitative approach for implementation);

- **Attributes:** name (associated name with the quality, e.g. precision) and level (associated to the degree in which the quality is provided);
- **Methods:**
- **Relationships:** is_aggregated_in *Context*

Element

- **Definition:** one specific object of the context (i.e. the name of a context information);
- **Examples:** user;
- **Attributes:** name (the name given to the properties of the contextual information) and description (a brief summary explaining the name of the element);
- **Methods:**
- **Relationships:** is_aggregated_in *Context*

Property

- **Definition:** one specific attribute that characterizes one element of the context;
- **Examples:** the birthdate of the user;
- **Attributes:** name (the name given to the properties of the contextual information) and description (a short explanation about the property of the element);
- **Methods:**
- **Relationships:** is_aggregated_in *Context*

3.1.3 Adaptation Core

The adaptation core involves the design decisions taken based on the processing of the contextual information available. This core includes inference and reasoning on top of the context in order to select and prioritize adaptation rules and their respective actions, completing a set of activities and functions performed to adapt some element of the interactive system;

Adaptation Process

- **Definition:** is the set of steps necessary to perform the adaptation, i.e. an adaptation lifecycle;
- **Examples:** given a specific context, the UI elements change, and are presented in a certain approach to the end user
- **Attributes:** id (a unique identifier associated to an adaptation process)
- **Methods:** start() (to begin the adaptation process), pause() (to temporarily terminate the process), and stop() (to terminate the process);
- **Relationships:** is_triggered_by an *Adapter*, is_composed_by one or more *AdaptationRules*

Adaptation Rule

- **Definition:** is a formal association connecting the context with the adaptation techniques, specifying how the system dynamically adapts according to the context [Koc00]. It can be structured according to the JECA approach [Nge07], defined as follows;
- **Examples:** if the user is dyslexic, then change the font type of the text;
- **Attributes:** id (a unique identifier associated to an adaptation rule), name (a unique name that characterizes the rule)

- **Methods:** calculatePriority() (to calculate the weight of the rule, based on context information provided) and apply() (to execute the rule in a given application);
- **Relationships:** composes *AdaptationProcess*, is_composed_by *Justification*, *Event*, *Condition*, and *Action*, and can be part_of one or several *Policy*

Justification

- **Definition:** a reason associated to the context, that provides a rationale, and aids to prioritize the adaptation and to justify with qualitative or quantitative information the selection of one specific action, it forms the reasoning context in which evaluation of the specific JECA rule to be performed [Nge07];
- **Examples:** there is no information available about the environment (then a default scenario must be considered);
- **Attributes:** id (a unique identifier for the justification), weight, priority, argument (associated values to support reasoning);
- **Methods:** check() (verifies whether there is a justification available for a given instance of the context);
- **Relationships:** composes an *AdaptationRule* and is_instantiated_by the *Context*

Event

- **Definition:** a specific status or change of status regarding the system, the user interaction or the context that supports specific actions;
- **Examples:** when the device is rotated;
- **Attributes:** id (a unique identifier to the event), name (a word or statement to qualitatively identify the event) and description (a brief description that characterizes the event);
- **Methods:** detect() (the event is listened and detected by the application);
- **Relationships:** composes an *AdaptationRule* and is_instantiated_by an instance of the *Context*

Condition

- **Definition:** an association between a given element and a given instance by means of an operator (e.g. equal, greater than) that enables comparison and evaluation (an enumeration named *OperatorType* provides possible instances for the operators);
- **Examples:** the user visual impairment is colour blindness;
- **Attributes:** id (a unique identifier for the given condition);
- **Methods:** evaluate() (function to check whether the condition is valid or not)
- **Relationships:** composes one or several *AdaptationRules* and it aggregates *Value*, *Operator* (specified by the enumeration *OperatorType*) and *ContextInformation*

Value

- **Definition:** the actual value that comes from the context of use, can be processed if needed (e.g. treated, converted), and verified according to the rule specification;
- **Examples:** 50;
- **Attributes:** name (a name associated to the value);
- **Methods:** process() (function to refine the value if needed, e.g. convert to a given unit or treat the information as necessary);

- **Relationships:** is_aggregated_with a *Condition* and instantiates a *Technique*

Operator

- **Definition:** the operator that permits a comparison between values;
- **Examples:** equal, greater than, different;
- **Attributes:** Type (an enumeration of possible instances is provided including equal, not_equal, and, or, less_than, greater_than, less_than_or_equal, greater_than_or_equal);
- **Methods:**
- **Relationships:** is_aggregated_with a *Condition* and is_related_to a *Technique*

ContextInformation

- **Definition:** an element of the context that can be retrieved and evaluated;
- **Examples:** the list of visual impairments associated to the given user;
- **Attributes:** id (a unique identifier associated to the contextual information of interest);
- **Methods:** retrieves() (function to associate a value with the element)
- **Relationships:** is_aggregated_with a *Condition*, is_related_to a *Technique* and is_instantiated_by the *Context*

Action

- **Definition:** a function that defines and activates the execution of the adaptation;
- **Examples:** change the font size to 12;
- **Attributes:** name (the name of the action);
- **Methods:** execute() (function that performs a given action) and cancel() (to interrupt the execution of the action);
- **Relationships:** composes one *AdaptationRule*, generalizes a *Technique*, is_presented_by a specific *PresentationMethod*, modifies a given *Model*, and aggregates a *Method*, a *Classifier*, a *Resource,* and a *Parameter*

Technique

- **Definition:** an operation that specifies a change in the system or in one or more properties of the system in order to adapt it;
- **Examples:** increase the font size;
- **Attributes:** id (a unique identifier associated with the technique), name (a name that characterizes the technique), reference (sources that define the technique, authors), description (a brief summary explaining it), rationale (steps needed to accomplish it), example (illustrative uses for the technique), context (context associated with it), advantages (qualities that are enhanced when applying it), disadvantages (qualities that are hindered while applying it), sample (a piece of code that implements it, e.g. an URL linking to web service that implements the technique), images (illustrative pictures of its application), observation (commentaries and notes about it), categories (tags to classify it);
- **Methods:**
- **Relationships:** is_associated_with *Value*, *Operator*, and *ContextInformation* and is generalized_by an *Action*

Policy

- **Definition:** an abstraction of a technique that governs it;
- **Examples:** if the user has low vision, but also a screen augmenter (as assistive device), there is no sense in augmenting the font size;
- **Attributes:** id (a unique identifier associated to the policy);
- **Methods:**
- **Relationships:** is_associated_with *AdaptationRules* and is_part_of *Strategy*;

Strategy

- **Definition:** an abstraction of a policy that governs it based on inferences performed with several contextual information;
- **Examples:** a combination of two or more given policies;
- **Attributes:** id (a unique identifier associated to the strategy);
- **Methods:** apply() (a function to activate a given strategy);
- **Relationships:** is_associated_with one or more given *Policy*

Method

- **Definition:** one specific function applied to change an element of the interactive system;
- **Examples:** re-size;
- **Attributes:** name (a given name associated with the method);
- **Methods:** execute() (function to apply a given method);
- **Relationships:** is_aggregated_with an *Action*

Classifier

- **Definition:** a definition of amount (subset, union, intersection or complement);
- **Examples:** all, any;
- **Attributes:** type (a given name that characterizes the classifier);
- **Methods:** set() (a function to associate a classifier with a given action);
- **Relationships:** is_aggregated_with an *Action*

Resource

- **Definition:** a component of the UI or the system that can be subject to adaptation, different granularity levels are considered, e.g. navigation, UI images, tables, their rows, columns or cells;
- **Examples:** image;
- **Attributes:** type (the name of the given resource defining the UI element);
- **Methods:** set() (a function to associate a given resource with an action)
- **Relationships:** is_aggregated_with an *Action*

Resource Property

- **Definition:** a specific characteristic or attribute of a resource;
- **Examples:** width of a table;
- **Attributes:** name (a characterization of the resource property);

- **Methods:**
- **Relationships:** belongs_to *Resource*

Parameter

- **Definition:** a value related to a given unit that specifies a parameter for the adaptation technique;
- **Examples:** +50%;
- **Attributes:** specification (a given value that characterizes the action);
- **Methods:** set() (a function to associate a given parameter to the action);
- **Relationships:** is_aggregated_with an *Action*

Presentation Method

- **Definition:** a explicit manner of presenting the adaptation to the end user aiming at avoiding disruption, possible types are listed as enumeration;
- **Examples:** an animation to present the re-sizing of an edit box;
- **Attributes:** id (a unique identifier associated with the presentation method), name (a short descriptive name associated with the presentation), and type (a characterization of the presentation);
- **Methods:** play() (a function to activate the execution of an presentation method), stop() (a function to stop the presentation method), pause() (a function to pause the presentation method), checkCompatibility() (a function to check whether the action is compatible with a given presentation method);
- **Relationships:** presents an *Action*;

Enumeration: possible presentation types include animation, brighten, blind, bounce, clip, cross fade, collapse, dim, drop, expand, explode, fade, fade in, fade out, fold, highlight, morph, plug in, plug out, progressive rendering, puf, pulsate, scale, self healing, shake, size, slide, spotlight, transfer.

3.1.4 Model-based Approach

An abstract representation of the reality (of the system, its different perspectives and the UI) that by means of reification or specialization is transformed from one abstraction level to another:

Model

- **Definition:** a formal definition of an interactive system, that can be decomposed in different abstraction levels, and complemented by different views, commonly expressed by means of a given notation (e.g. UML, XML, CTT);
- **Examples:** a UsiXML model specifying an interactive system;
- **Attributes:** id (a unique identifier of the model) and a description (the model definition);
- **Methods:** reify() (an specialization of a model to make it more concrete) and abstract() (transformation to a higher abstraction level);
- **Relationships:** is_composed_by one or several models of a *Task*, *AUI*, *CUI* and *FUI* and is_modified_by an *Action*

Task

- **Definition:** a set of actions and activities to be executed according to given constraints, as ordering, to achieve a specific goal while interacting with the system;
- **Examples:** an CTT or an HTA task tree;

- **Attributes:** id (a unique identifier associated to the task tree) and description (a definition of the task tree: nodes, relationships, properties, etc.);
- **Methods:** reify() (function to transform a task tree into an AUI model);
- **Relationships:** composes one or several *Models*

AUI (Abstract User Interface)

- **Definition:** the abstract definition of the system and its UI that is domain and platform-independent;
- **Examples:** a UsiXML model expressing an AUI model;
- **Attributes:** id (a unique identifier associated to the AUI model) and description (a definition of the AUI components);
- **Methods:** reify() (function to transform an AUI model into a more concrete definition, i.e. a CUI model) and abstract() (function to transform an AUI model into a more abstract definition, i.e. a Task Tree);
- **Relationships:** composes one or several *Models*

CUI (Concrete User Interface)

- **Definition:** a more concrete definition of the system, its UI and its components;
- **Examples:** a UsiXML model expressing a CUI model;
- **Attributes:** id (a unique identifier associated to the CUI model) and description (a definition of the CUI components);
- **Methods:** reify() (function to transform an CUI model into a more concrete definition, i.e. a FUI model) and abstract() (function to transform an CUI model into a more abstract definition, i.e. an AUI model);
- **Relationships:** composes one or several *Models*

FUI (Final User Interface)

- **Definition:** the (graphical) user interface to be presented and/or rendered to the end user;
- **Examples:** a running application;
- **Attributes:** id (a unique identifier associated to the FUI model) and description (a definition of the FUI components);
- **Methods:** abstract() (function to transform an FUI model into a more abstract definition, i.e. an CUI model);
- **Relationships:** composes one or several *Models*

3.1.5 Discussion

It is challenging to define an abstract model that clearly represents an information system, mainly because there are several approaches that are alternative solutions and provide base to generate the same instantiations. For CAMM we opted to model CAA in a high abstraction level, ensuring technology-independency, context-independency (regardless of user, platform and environment) and domain-independency. The concepts covered have been identified based on extensive literature search, aiming at covering the complete adaptation lifecycle. Although CAMM provides a unified terminology for the CAA domain, only by ensuring its broad adoption, the benefits of consistency could be felt, e.g. compatible results, extensible applications, consistent documentation and improved communication among stakeholders.

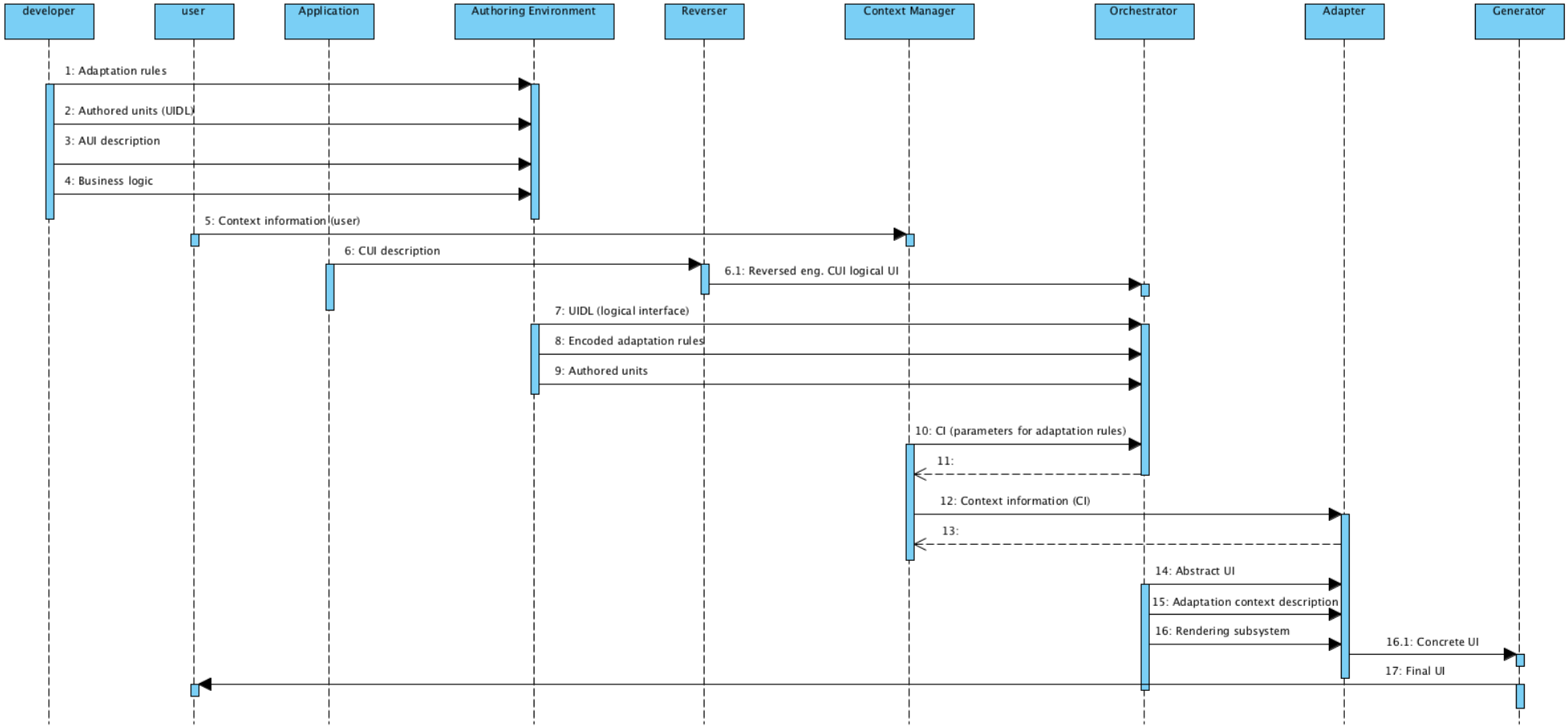

Figure 4. Sequence diagram: an overview of the adaptation process)

4 UML Models

The UML models represent an interactive system from different perspectives. In this section we target at context-aware adaptation processes, focusing on: (i) the sequence of messages exchanged between Serenoa modules, by means of a Sequence diagram, (ii) the phases of an adaptation process, by means of a Statechart diagram, and (iii) the potential adaptation techniques of an adaptable or adaptive system, represented by means of Use Case diagrams.

4.1 Sequence Diagram

The Sequence diagram illustrates the main modules that compose the Serenoa framework, including: agents (developers and users), application, authoring environment (e.g. Quill), reverser, context manager, orchestrator, adapter and generator. A set of 17 messages is presented. They compose the core communication needed for an adaptation process, starting from the rules proposed by the developers and finishing by the presentation of the adapted UI for the end user. While Figure 4 presents an overview of the adaptation communication, Figure 5 illustrates the messages exchanged within the Context Manager. Such definitions provide a high level view of the process, further details are presented in the deliverables 1.2.1, 2.2.1 and 3.3.1.

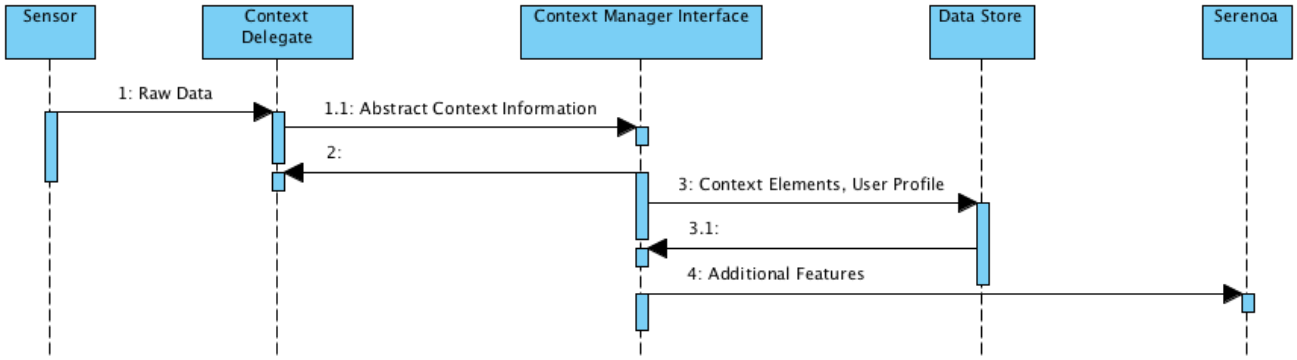

Figure 5. Sequence diagram: the Context Manager

4.2 State chart Diagram

A state chart diagram covers the different phases of a process. In Figure 6 we illustrate an adaptation process covering the ISATINE [Lóp09] framework and also the mental model of Norman. We complement such definition by involving also the end user participation in the adaptation process, i.e. by evaluating the adapted UI, he or she decided whether the adaptation can be finalized (satisfied with the results) or no.

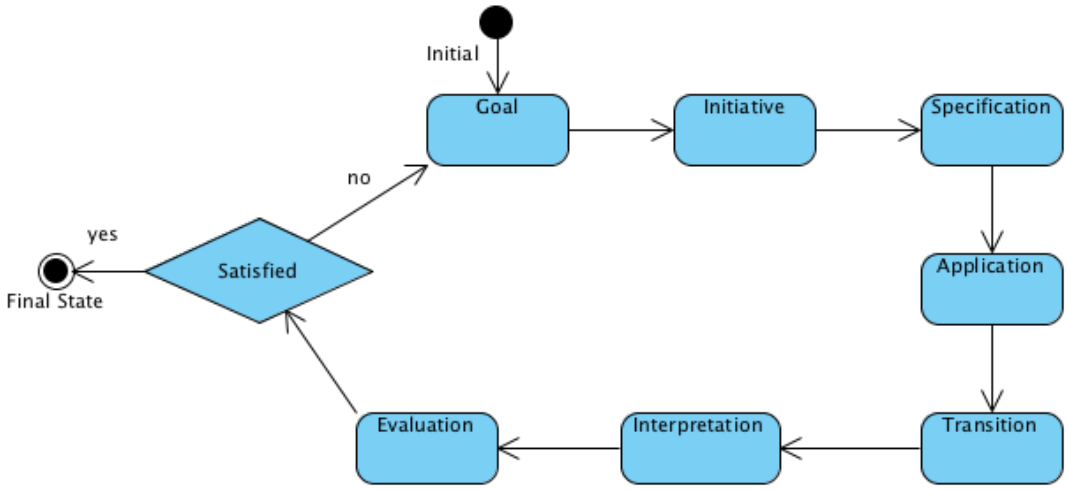

Figure 6. State chart diagram (GISATIE) [Lóp09]

In a complementary perspective, the adaptation process can be summarized in 4 main phases: starting from context gathering, passing through adaptation rules, to the transformations of the UI's in a model-based approach, and finishing with the evaluation of the adaptation results by the end user. Such process is cyclic, in a sense that, if the end user is not satisfied, new adaptations must be proposed and performed, until the results are satisfactory.

4.3 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 the second release they were updated accommodating some extensions and further techniques for adaptation. This third release presents a consolidated version of the use cases, with additional techniques and synchronized with the complete version of CARF. 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. The use cases are organized according to the resource that is subject to the adaptation (i.e. content, navigation and presentation), because this classification is less ambiguous than other criteria, as context information or software qualities. Still some overlapping is expected, mainly between techniques that affect presentation and contents.

Figure 7 illustrates a general view of the use case diagram. 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, in a mixed-approach for adaptation. Other specializations, such as developers, or designers can be also considered [Gan07]. 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.

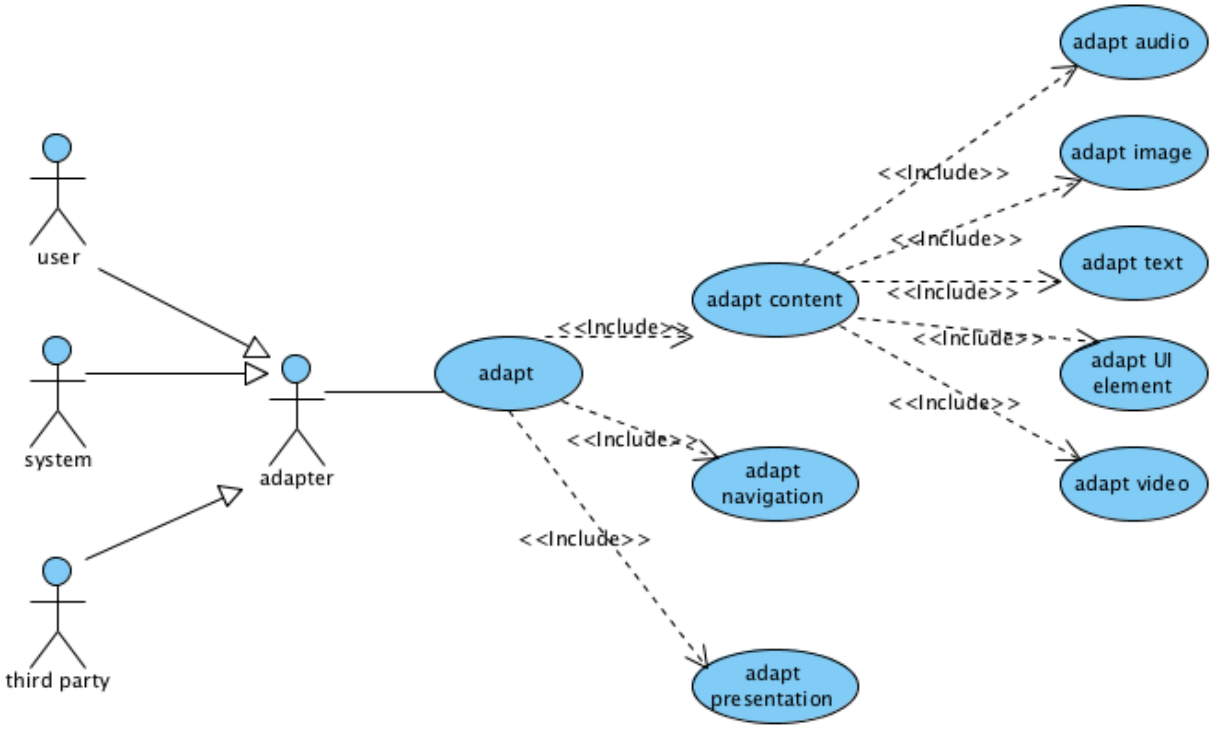

Figure 7. Use Cases Overview: the main agent, named adapter, can be specialized as an end user, a system or a third party. It is responsible for adapting: contents (audio, image, text, UI elements, or video), navigation or presentation.

Each of these adaptations have their specific use cases, they were detailed in further diagrams: Figure 8 illustrates the adaptations for contents in general, Figure 9 for audio contents, Figure 10 for images, Figure 11 for text, Figure 12 for UI elements and Figure 13, for video. The use cases regarding adaptation of navigation and presentations are respectively illustrated by Figure 14 and by Figure 15.

Concerning the adaptation of contents, in general, without any specific format, 12 generic use cases were considered, and 8 possible extensions, refining the techniques of change, re-sizing and re-ordering according to possible specializations (e.g. re-size the height, or re-order by chronological order). These use cases use are listed in Figure 8.

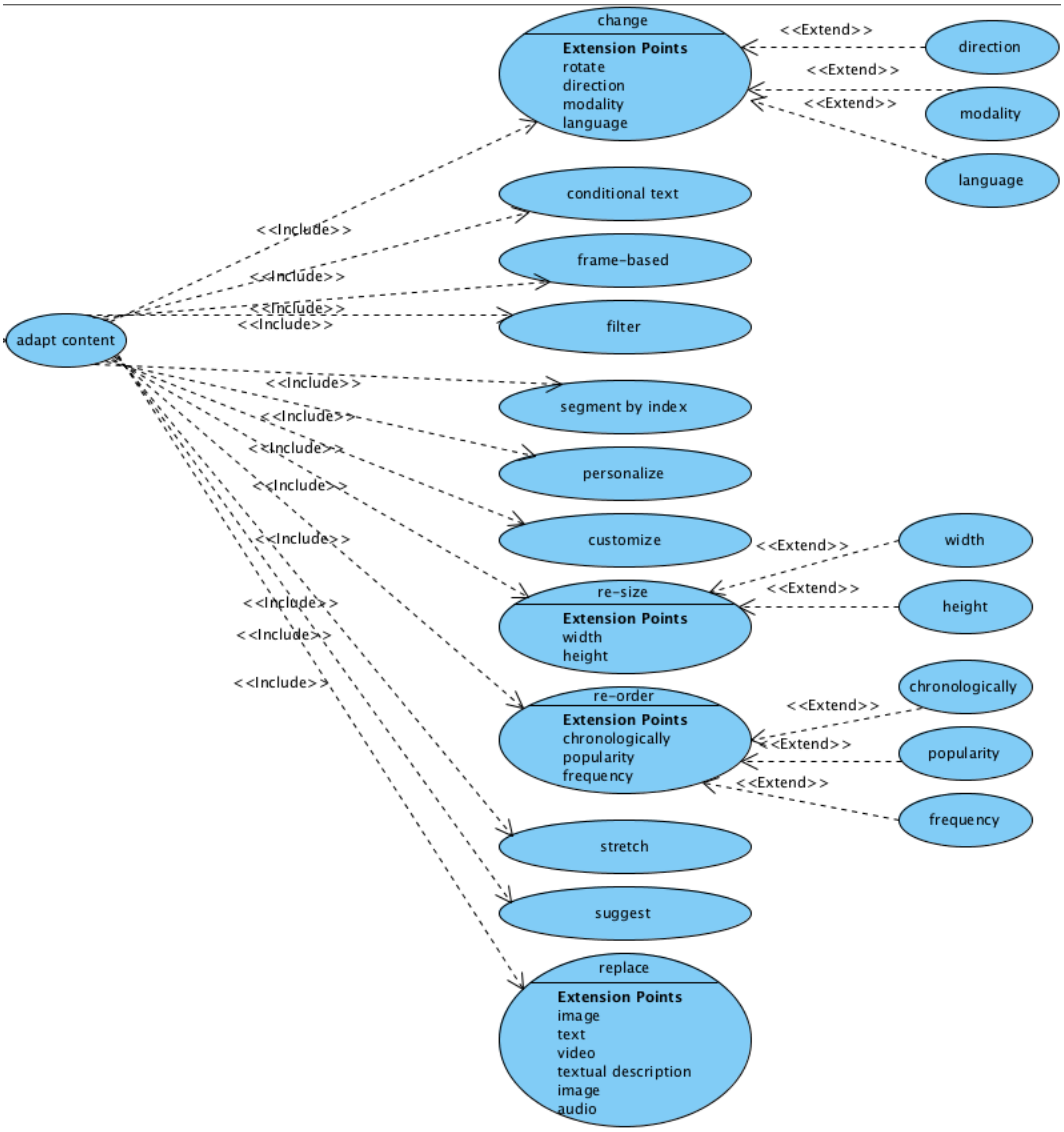

Figure 8. Use Case Diagram for Adapting Content

Concerning the adaptation of audio contents, as Figure 9 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 or by content) the audio. 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 if the noise level is too high, it must be replaced by an equivalent content in another modality, e.g. an image if there is a screen available to present it.

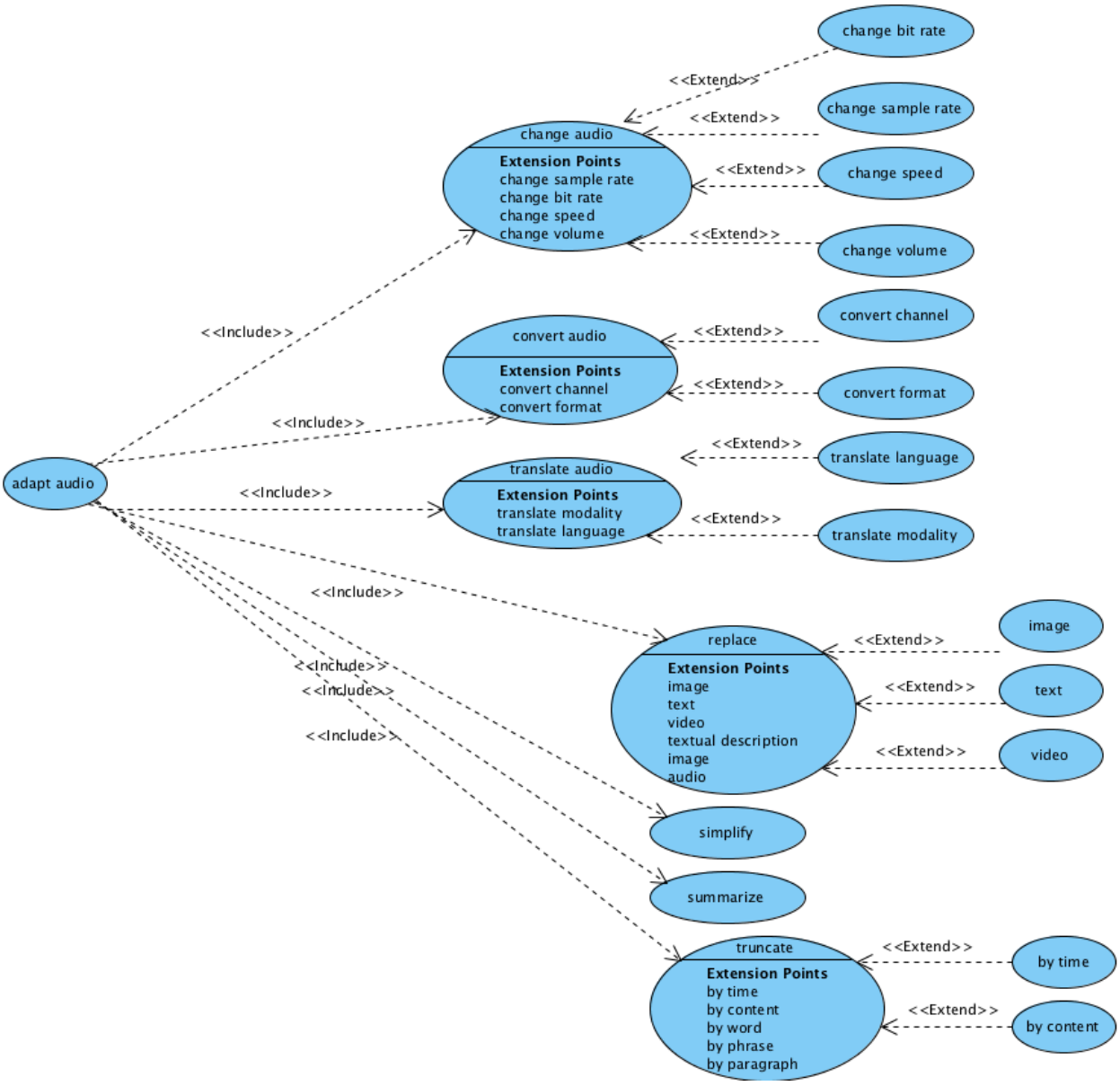

Figure 9. Use Case Diagram for Adapting Audio

The Figure 10 illustrates the Use Case Diagram for adapting images. 22 techniques are included and they are refined in 16 extensions. These use cases involve changes that may affect the complete image, or its specific properties, such as their dimension, or specific colours' properties (e.g. transparency level).

Figure 11 illustrates the Use Case Diagram for adaptation of text. 18 techniques were considered, and 28 extensions. This UCD composes an extensive list of possibilities in terms of context-aware adaptation for textual contents.

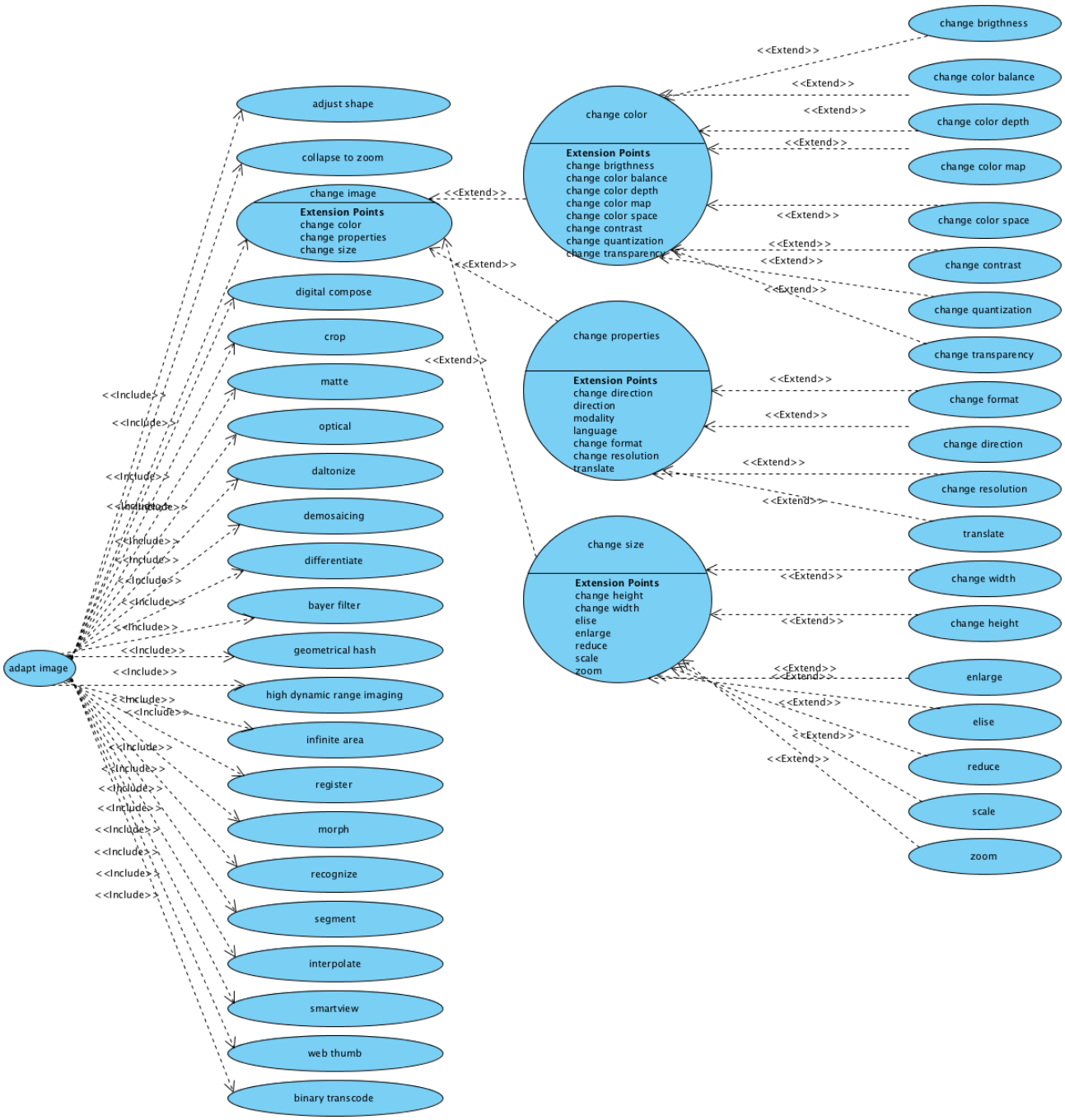

Figure 10. Use Case Diagram for Adapting Image

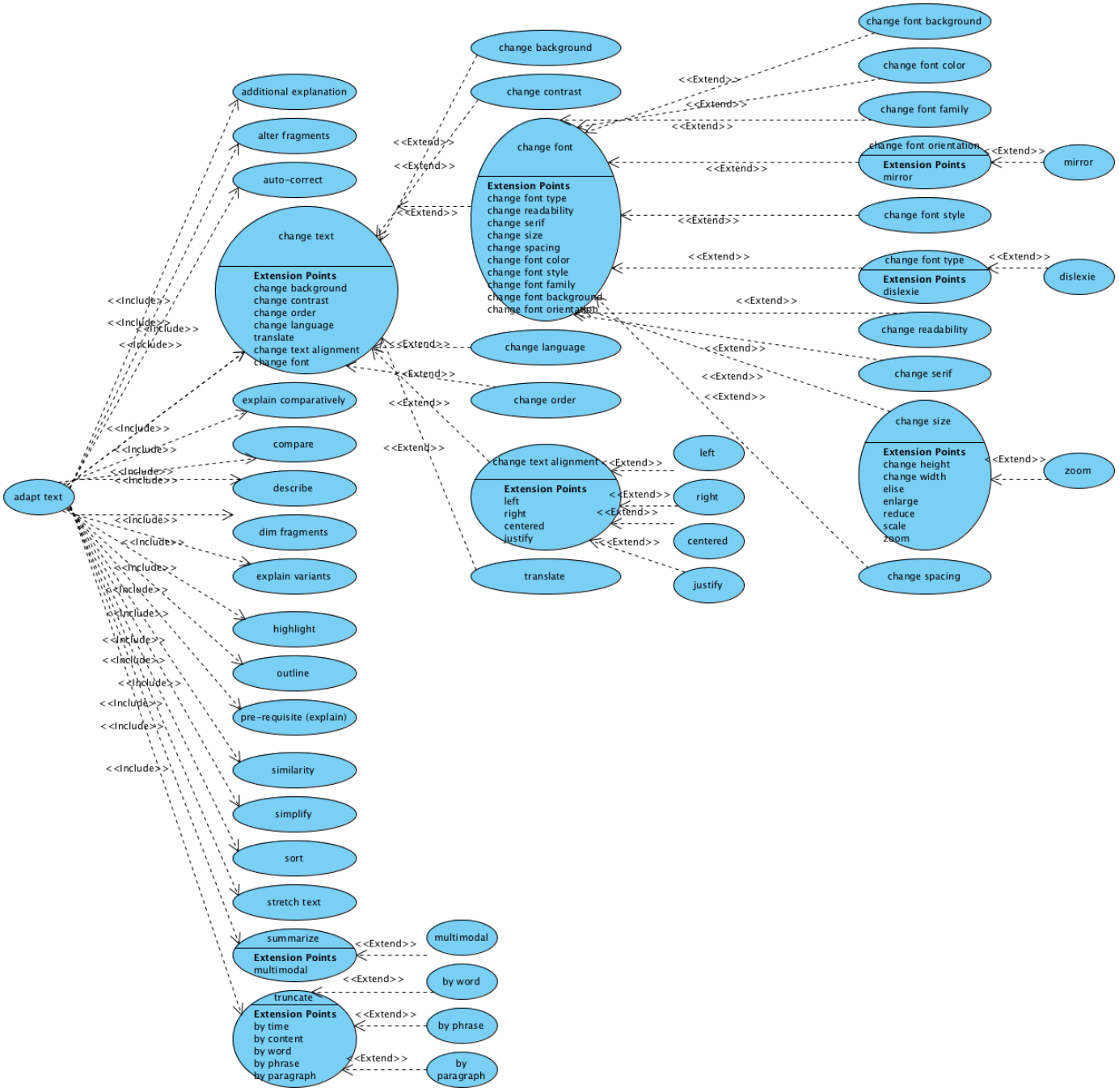

Figure 11. Use Case Diagram for Adapting Text

Figure 12 illustrates the Use Case Diagram for UI elements. 15 techniques are included and 6 possible extensions; they involve different types of widgets, such as forms, text boxes and tables.

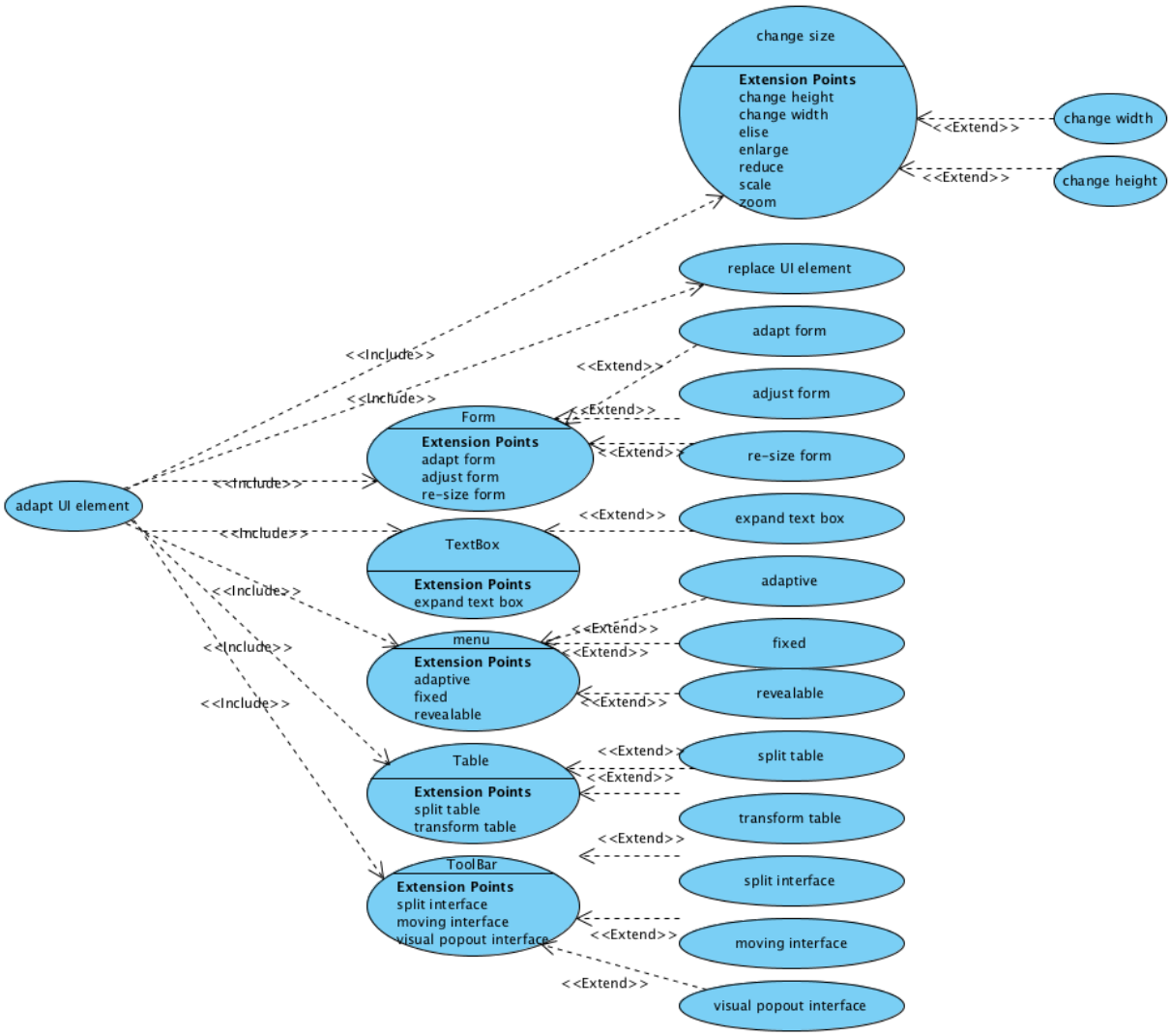

Figure 12. Use Case Diagram for Adapting UI Elements

Figure 13 illustrates the Use Case Diagram for adaptation of video. It includes a set of 9 techniques that are extended by 14 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).

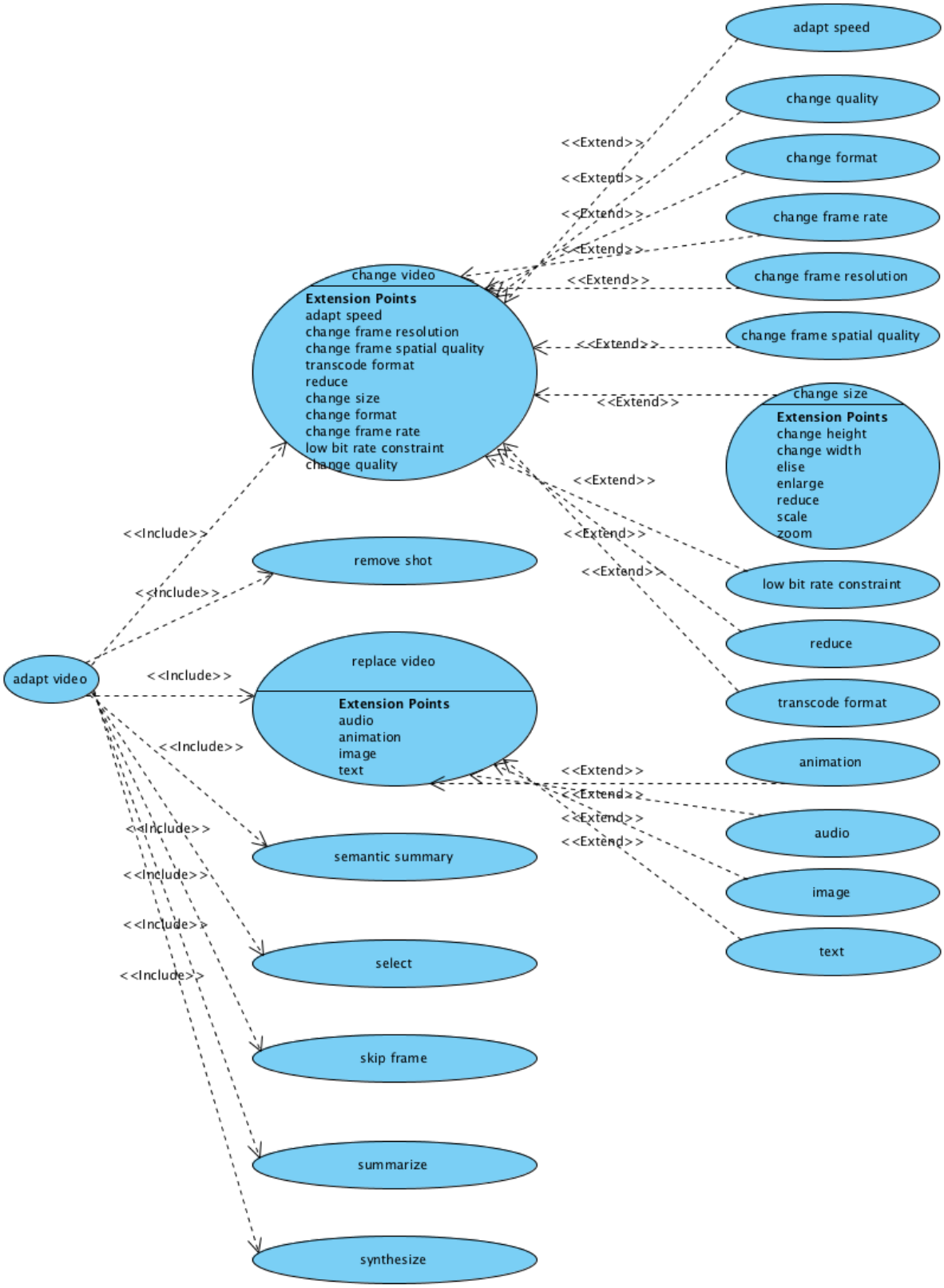

Figure 13. Use Case Diagram for Adapting Video Contents

Figure 14 illustrates the Use Case Diagram for adaptation of Navigation. It includes 31 adaptation techniques and 6 possible extensions.

Figure 15 illustrates the Use Case Diagram for adaptation of Presentation. It is composed by 23 techniques and 8 extensions.

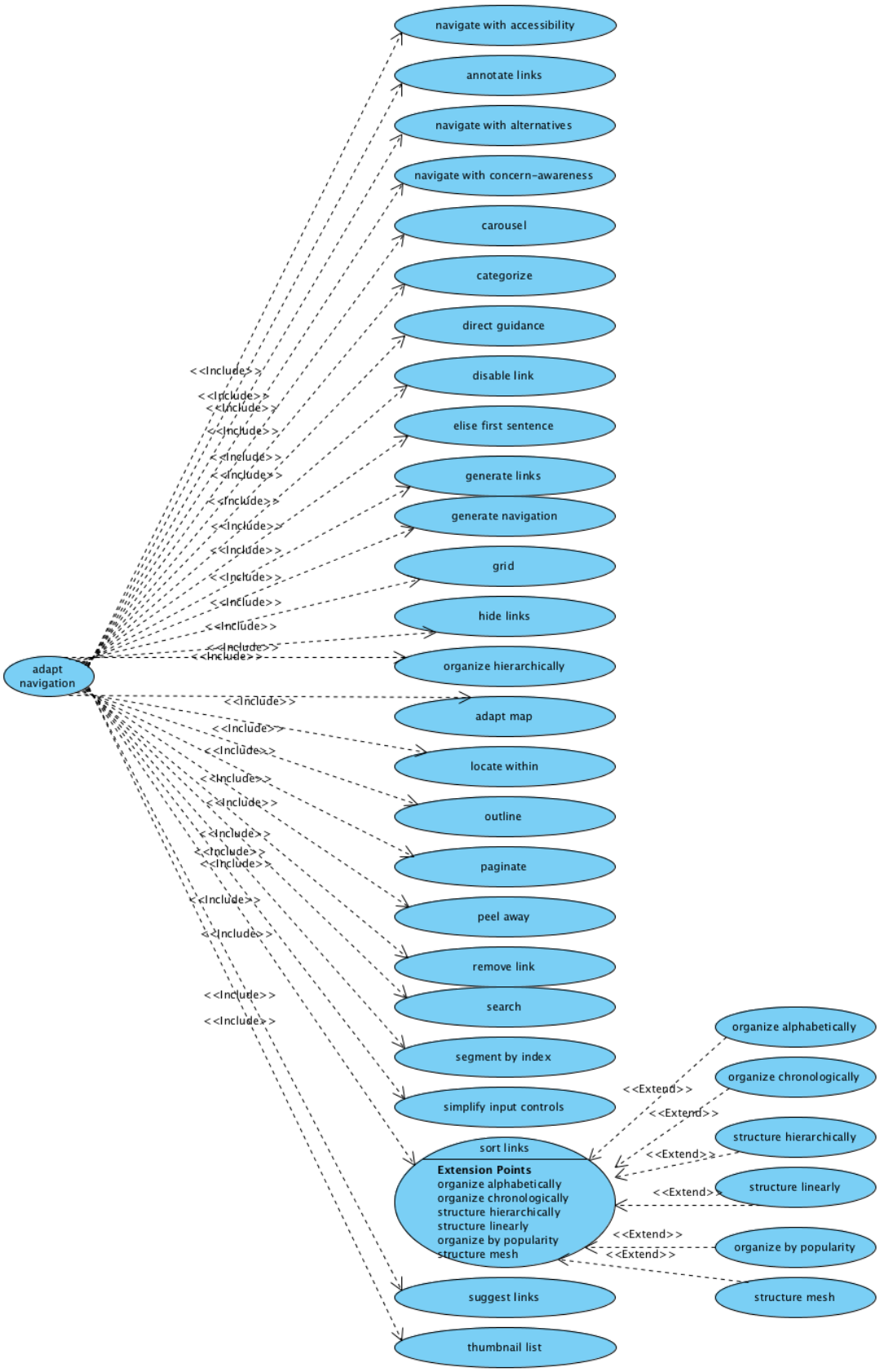

Figure 14. Use Case Diagram for Adapting the Navigation

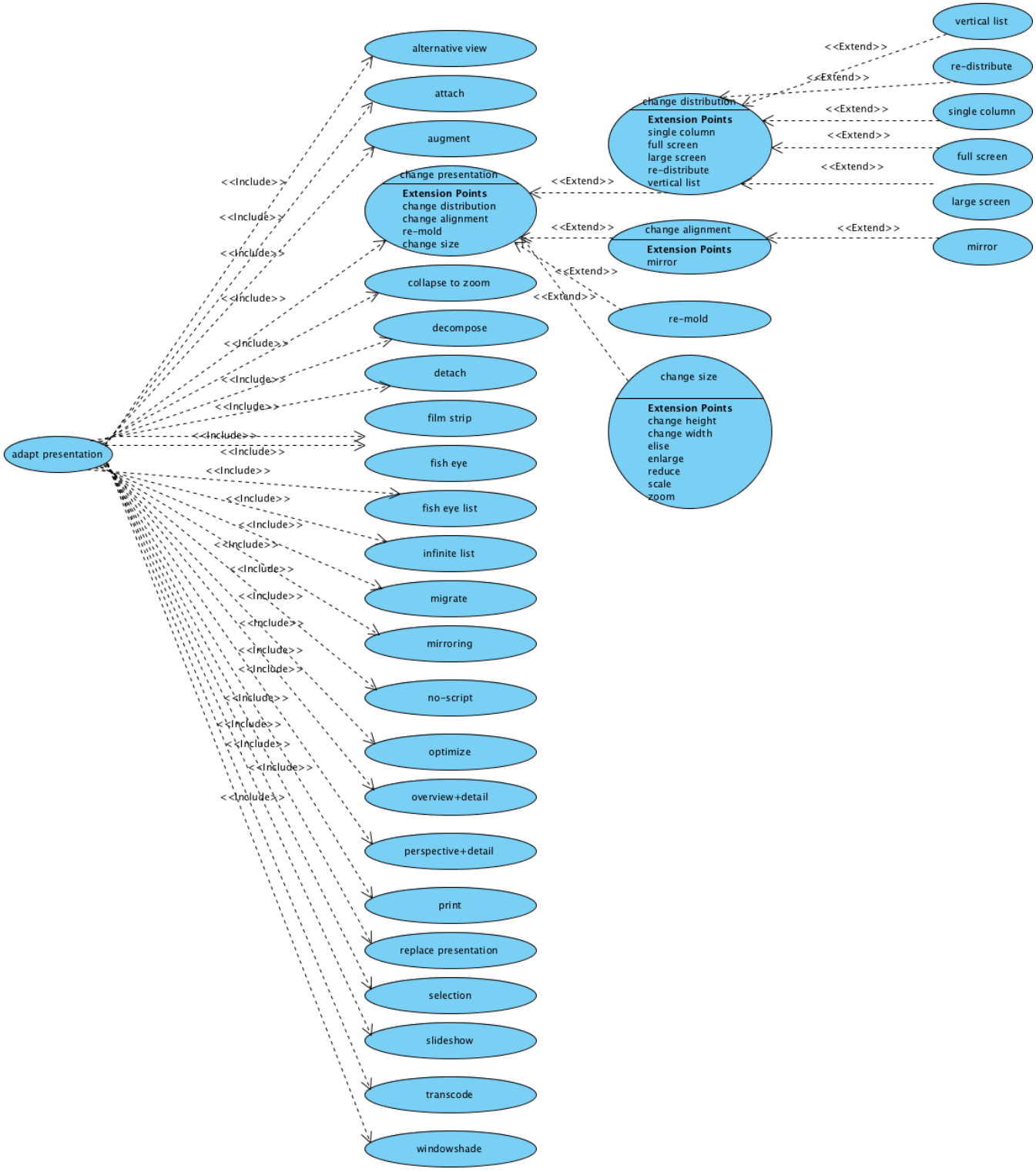

Figure 15. Use Case Diagram for Adapting the Presentation

5 Final Remarks

This deliverable consolidates the previous works on the reference models of Serenoa. To do so, we begin by briefly presenting and discussing a set of related works targeted at defining meta-models for context-aware adaptation and then we follow by presenting the consolidated versions of the reference models for CAA.

This deliverable presents an updated version of CAMM, a meta-model for CAA and also a set of UML diagrams, including a sequence diagram, a state chart and use cases that cover the updated list of adaptation techniques.

5.1 Discussion

Although the work presented in this deliverable extensively considers CAA, covering several concepts and techniques for supporting context-aware adaptation, it is not possible to be exhaustive. In this sense, by adopting well-known standard and notations, as MOF and UML, for the reference models of Serenoa, we enable the continuous update and extension of the diagrams, aiming to synchronize them with necessary refinements according to the future developments in the domain of interest.

5.2 Conclusion

The modelling of CAA provides several benefits for stakeholders, among which we can highlight:

- unified development approaches
- a consistent terminology
- support and guidance during the development phases
- a graphical representation of main concepts, as well as their relationships and properties
- an overview of how context-aware adaptation is defined
- a common ground for several instantiations.

5.3 Future Work

The models presented in this deliverable can be refined and also extended. The MOF description adopted for CAMM can lead to a consistent vocabulary and an adaptation language. It can be instantiated by different applications, regardless of domain, context, platforms or technology. In order to synchronize the current definitions with future progress in this domain, the models proposed in this deliverable should be subject to continuous extensions.

References

P. Brusilovsky and D. W. Cooper, “Domain, task, and user models for an adaptive hypermedia performance support system,” In: Proc. of the 7th int. conference on Intelligent user interfaces. ACM, p. 23-30, 2002

G. Calvary, O. Daassi, J. Coutaz, and A. Demeure, “Des widgets aux comets pour la Plasticité des Systèmes Interactifs,” Revue d'Interaction Homme-Machine, v. 6, n. 1, p. 33-53, 2005.

Ch.-E. Dessart, V. G. Motti and J. Vanderdonckt, J, “Showing User Interface Adaptivity by Animated Transitions” In Proc. of 3rd ACM Symp. on Eng. Interactive Comp. Sys. EICS’2011. ACM, NY, 95-104.

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.

Fahrmair, M., Sitou, W., Spanfelner, B. 2005. An engineering approach to adaptation and calibration. In Proceedings of the Second international conference on Modeling and Retrieval of Context (MRC'05), Thomas R. Roth-Berghofer, Stefan Schulz, and David B. Leake (Eds.). Springer-Verlag, Berlin, Heidelberg, 134-147. DOI=10.1007/11740674_9 http://dx.doi.org/10.1007/11740674_9

C. R. G. de Farias, M. M. Leite, C. Z. Calvi, R. M. Pessoa, and J. G. Pereira Filho, “A MOF metamodel for the development of context-aware mobile applications,” In: Proceedings of the 2007 ACM symposium on Applied computing. ACM, 2007. p. 947-952.

F. Fuchs, I. Hochstatter, M. Krause, and M. Berger, “A metamodel approach to context information,” In: Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on. IEEE, 2005. p. 8-14.

V. Ganneau, G. Calvary, and R. Demumieux, “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. ACM, p. 91-98, 2007.

Horvitz, E.: Principles of Mixed-Initiative User Interfaces. Proc. of ACM Conf. on Human Aspects in Computing Systems CHI 1999, ACM Press, New York, 1999, pp. 159-166.

Kappel, T.G., Retschitzegger, W., Kimmerstorfer, E., Proll B., Schwinger, W., Hofer, T. Towards a Generic Customisation Model for Ubiquitous Web Applications. *In Proceedings of the Second Int. Workshop on Web-Oriented Software Technology (IWWOST’02)*, 79-104, Malaga, Spain, June 2002. ISBN: 84-931538-9-3.

Kobsa A. 2007, Generic User Modeling Systems. *In The Adaptive Web (LNCS)*, Vol. 4321, 136-154.

N. P. D. Koch, “Software engineering for adaptive hypermedia systems and development process,” 2000. PhD Thesis

V. López-Jaquero, J. Vanderdonckt, F. Montero, P. González, “Towards an extended model of user interface adaptation: the ISATINE framework,” In: Engineering Interactive Systems. Springer Berlin Heidelberg, 2008. p. 374-392.

Limbourg, Quentin, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, and Víctor López-Jaquero. "USIXML: A language supporting multi-path development of user interfaces." In Engineering human computer interaction and interactive systems, pp. 200-220. Springer Berlin Heidelberg, 2005.

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 Project, (2012) Context of Use Meta model. Available online at: http://forge.morfeo-project.org/wiki_en/index.php/Context_Of_Use_ Metamodel#Introduction.

Motti, V. G., Vanderdonckt, J. A Computational Framework for Context-aware Adaptation of User Interfaces, In Proc. of the Seventh IEEE International Conference on Research Challenges in Information Science (2013)

Nebeling, M. Context-Aware and Adaptive Web Applications: A Crowdsourcing Approach, *In Proc of ICWE*, 2011.

Ngeow, Y.C., Mustapha, A.K., Goh, E., Low, H.K.: Context-aware Workflow Management Engine for Networked Devices. International Journal of Multimedia and Ubiquitous Engineering (IJMUE) 2(3), 33-47 (2007).

Norman, D.A.:1986, Cognitive Engineering. In: Norman, D.A., Draper, S.W. (Eds.): User Centered System Design. Lawrence Erlbaum Associates, Hillsdale, pp. 31–61.

Sottet, Jean-Sébastien, Vincent Ganneau, Gaëlle Calvary, Joëlle Coutaz, Alexandre Demeure, Jean-Marie Favre, and Rachel Demumieux. "Model-driven adaptation for plastic user interfaces." In Human-Computer Interaction–INTERACT 2007, pp. 397-410. Springer Berlin Heidelberg, 2007.

UsiXML Specification. Available at: usixml.org

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).

de Vrieze, Paul, Patrick van Bommel, and Theo van der Weide. "A generic adaptivity model in adaptive hypermedia." In Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 344-347. Springer Berlin Heidelberg, 2004.

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/