State-of-the-art of the User, Device and Application Models

Deliverable 5.1 :: Public

Keywords: user models, application models, device models, CC/PP, UAProf, WAI-ARIA, UML

Inclusive Future Internet Web Services

Table of Contents

List of Figures

List of Tables

Contractual Date of Delivery to the EC:		30th April 2011 + 60 days	
Actual Date of Delivery to the EC:		17th May 2011	
Editor:		Carlos A Velasco (FhG/FIT)	
Contributors:		I2Web Consortium	
DOCUMENT HISTORY			
Version	Version date	Responsible	Description
0.1	4th April 2011	FIT	Initial ToC and descriptions.
0.5	18th April 2011	FIT	First initial draft.
1.0	30th April 2011	FIT	Partner contributions edited and merged.
1.1	17th May 2011	FIT	Final version.

Executive Summary

One of the key components of the I2Web architecture is the modelling of the user, device and application interaction. In this document we review their state-of-the-art and evaluate possible approaches and developments for the project.

1 Introduction

The objective of WP5 is to create all the models that are going to be integrated into the components of the I2Web project developed in WP6 and WP7. These models are based on state-of-the-art semantic models, extended and refined with the input from WP3 and the results of WP4. This document provides a first overview of existing models for the three key areas of the project: users, devices and applications.

It is known that most of the modelling approaches are based upon common technologies, thus before diving into them, the following chapter describes some of this underlying techniques.

2 Underlying technologies

Modelling frameworks have several technologies in common like RDF and UML. To avoid repetitions in the following sections, we will summarise them in this chapter. We assume these technologies are common knowledge and thus we will provide only a quick description and references to the specification.

2.1 Resource Description Framework (RDF)

The Resource Description Framework [rdf2004] is a general-purpose language for representing information in the Web and is the underlying technology for the semantic web. RDF provides the basis for describing technology in a device independent, well-formed way, permitting its interpretation by a machine. A complete description of RDF is beyond the scope of this report.

RDF provides means to describe resources present in a networked environment and their inter-relationships. These relationships are described through a graph model which connects the subject of interest with attributes or properties related to it. For each resource subject, an object is related through a predicate. This relationship can be represented through a simple directed acyclic graph (DAG, see Figure 1).

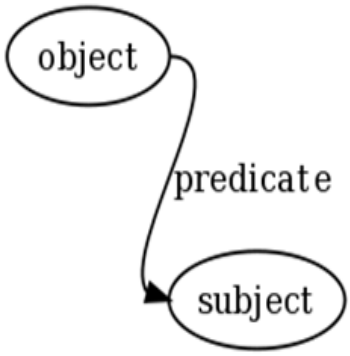

Figure 1. The RDF DAG.

Through this structure a wide variety of direct relationships can be represented and, due to the transitive properties of these DAG models, indirect relationships can be inferred. In order to demonstrate how such a model could be used in representing the collected attributes of a device, consider a single line Braille display (SLBD) sold by the company DisplayCorp. Consider the following set of English sentences that describe this device:

- SLBD *has a company* named **DisplayCorp**.
- SLBD *has a character set* of **8-dot Braille cells**.
- SLBD *has a number of cells* of **60 cells**.

Taking these statements as facts, the information can be restructured as an RDF model of the characteristic relationships of the SLBD. Assuming that the subject of interest is the SLBD, and the predicates are presented in italics, with the objects being presented in bold, the following RDF DAG in Figure 2 can be constructed:

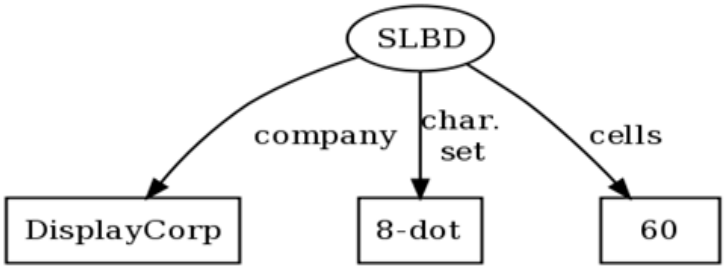

Figure 2. A sample RDF graph model for a single line Braille display.

At this point, the information is no more machine readable than it was in the original English language description. However, each piece of relationship information has to be incorporated into a web resource, described through a Uniform Resource Identifier (URI), which in turn can be used as a machine readable reference for the resource. As such, the subject, relationship and object information could be substituted with the URI references in the RDF model. Continuing the above example, if it is assumed that that the SLBD has a web page on the DisplayCorp the following substitutions could be made:

- Subjects
  - SLBD – http://www.displaycorp.com/slbd
- Relationships (Predicates)
  - cell number – http://www.displaycorp.com/terms/cell
  - company – http://www.displaycorp.com/terms/company
  - character set – http://www.displaycorp.com/terms/charset

For simplicity in this example, the objects have not been replaced with URI references and are left as atomic data attributes. The resulting RDF model is presented in Figure 3.

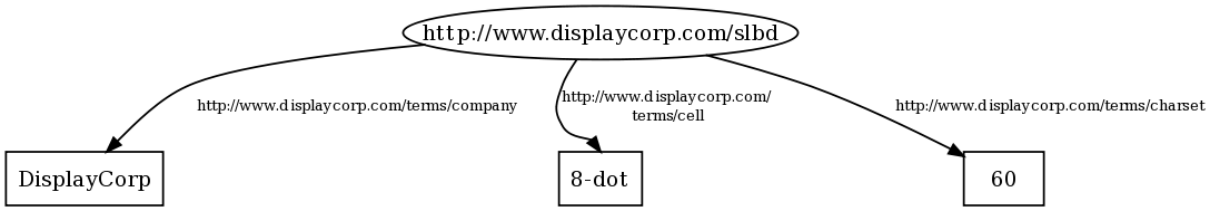

Figure 3. A sample RDF model with URI references.

In this way, a full machine-readable model of the Braille display was created. A complete description of semantic web standards can be found at: http://www.w3.org/RDF/.

2.2 Composite Capabilities/Preference Profiles (CC/PP)

In this section we will discuss the use of Composite Capabilities/Preferences Profile (CC/PP) [ccpp20] in depicting the user or device preferences and profiles to be used in I2Web. The descriptive power of RDF provides the ability to represent information about resources on the World Wide Web. The resources that are relevant to I2Web are the preferences of the devices and of users when they access our services.

The Composite Capability/Preferences Profile (CC/PP) is an RDF based format that can be used to represent a wide array of devices capabilities and user preferences. The vocabulary

evolved from its first version [ccpp10] to be aligned with the latest version of RDF and UAProf. However, this second version never reached the Recommendation status [ccpp20].

This framework is flexible and extensible, allowing the definition of a wide variety of hardware technologies, software technologies, user agent technologies and user preferences and characteristics that will be available to the I2Web services. A model based upon this framework could provide the ability to detect devices that are being used to access the system and then automatically adapt content to meet the needs and preferences of the user [velasco2004].

The general structure of a CC/PP client profile is a two-level tree: components and attributes. This simplicity makes it ideal to the purpose of representing user and device models. Also using the same underlying framework for both models improves their interoperability [ccpp20]:

- **Components**
  A CC/PP profile contains one or more components, and each component contains one or more attributes. Each component is represented by a resource of type ccpp:Component, and related to the client profile resource by a ccpp:component property.[1]
- **Attributes**
  The data model represents CC/PP attributes as named properties linking a subject resource to an associated object resource or RDF typed literal value. To describe capabilities and preferences, the client being described is a resource whose features are described by labeled graph edges from that resource to corresponding object values. The graph edge labels identify the client feature (CC/PP attribute) being described, and the corresponding object values are the feature values.

As an example, a profile could describe the capabilities of the technology of a user. Examples of such components could be:

1. Hardware: The hardware platform on which any software will run.
2. Software: The software platform providing hosting for the applications.
3. Browser: An application used to access information such as a web browser.

These components can then refined through the definition of attributes to further describe their capabilities. These attributes represent the properties that need to be transmitted to any service wishing to deliver information through those components.

Returning to the example in the following section, the SLBD can now be represented in the CC/PP tree. This device could be categorized in our example components as a hardware device. Adding to the example, assume that the user also uses a screen reader. This screen reader has a name, a version, a voice type, and a vendor. Furthermore, in order to use this hardware/software to access the web, a web browser must be defined. An example configuration of this tree is represented in Figure 4:

[1] The ccpp namespace is http://www.w3.org/2006/09/20-ccpp-schema#

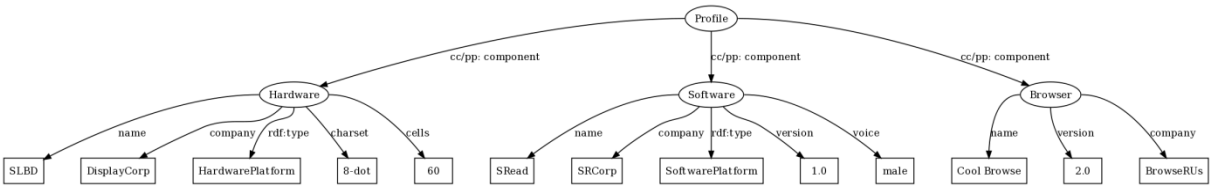

Figure 4. A sample CC/PP profile for a single line Braille display with a screen reader and web browser.

Clearly this is a contrived example for demonstration purposes. The real world complexities of devices and the software that runs on them are much higher. As a result there is a need to standardise the components and attributes that are going to be used within any given CC/PP implementation. To provide a standard description, a vocabulary can be defined for use as an RDF schema. For each vocabulary, each resource is defined as having a set of required CC/PP attributes. These attributes can be of any of the types defined in the RDF schema.

2.2.1 Extending the Schema/Vocabulary

One of the reasons for CC/PP being a candidate for user and device modelling in our architecture, is the extensible power of the specification. Through the use of vocabularies it is possible to define new attribute types, allowing the specialisation of device descriptions to better support the assistive technologies and user agents used by people with disabilities.

There are a number of CC/PP implementations to date. Below is a listing of those which have been reported to the W3C CC/PP working group. Unfortunately, not all are available outside the companies or products in which they are used, and sometimes they may be part of discontinued products. Describing their features is outside the scope of this document.

Name	Originator	URI	Reported to working group	Open Source
Musashi	Ericsson Wasalab	http://www.w3.org/Mobile/CCPP/implday/#demo1	November 15, 2000	No
WAP Application Server	Ericsson	http://www.w3.org/Mobile/CCPP/implday/#demo1	November 15, 2000	No
Panda/Sasa	Kiniko Yasuda, Keio University	http://yax.tom.sfc.keio.ac.jp/panda/slidemaker/0011ccpp/Overview.html	November 15, 2001	Yes
SBC/TRI Reference implementation	SBC/TRI	http://www.w3.org/Mobile/CCPP/implday/#demo1	November 15, 2001	No
Information Architects	Chris Woodrow, Information Architects	http://www.w3.org/Mobile/CCPP/implday/#demo1	November 15, 2001	No
W3C	Jigsaw Team	http://www.w3.org/Jigsaw/	Nov. 15, 2000	Yes
DELI	Mark Butler, Hewlett	http://www.hpl.hp.com/techreports/2001/HPL-2001-	Nov. 2, 2001	Yes

Name	Originator	URI	Reported to working group	Open Source
	Packard Laboratories	260.html		

Table 1. List of old CC/PP implementations.

Notice that the listed implementations are quite old. This is due to the fact at the moment the industry of mobile devices is going through a strong competitive phase where the interest in standards development for device adaptation plays a secondary role. However, from our point of view CC/PP is a quite mature framework for our objectives.

3 User modelling

3.1 The notion of user model

User modelling is the creation of different personas in order to make decisions before actual events occur. It is a cross-disciplinary analysis of how humans interact within specific computer environments. Some researchers in the field of Human-Computer Interaction (HCI) have been interested in user modelling because of its potential to improve the nature of HCI systems. Understanding how users will behave can assist in building better websites and software applications, especially for a wider array of users, including those with disabilities. The primary purpose of user modelling is to understand how users with different attributes are likely to interact with a user interface. Thus, the main imperative of user modelling is that it has to be done for the benefit of users. Lately, a lot has been said about this, but in reality progress has been slow and difficult, and success stories rare [gfisch2001].

User models are defined as images of users as they appear in systems. Models reside inside a computer environment. To address a broader class of user-adaptive systems, a user model can be defined as a "… knowledge source in an intelligent system which contains assumptions on different aspects of the user that may be relevant to the system's adaptive behaviour. These assumptions must be separable from the rest of system's knowledge." [sosnov2007] Consequently, user modelling is the field of information science dealing with elicitation, representation and utilization of user models.

User models are created in different ways for adaptive systems and adaptable systems. An adaptable system is one over which the user is given some control. This is done through preferences or customizable elements. User modelling in an adaptable system is both performed in advance to create useful choices and ongoing as users take advantage of those choices. Data can be collected on the choices users make to guide further development. A simple example is a system with both 'Basic' and 'Advanced' interface that allows the user to choose in which way to interact. Adaptive systems are those in which the interface and/or content are structured to adapt to the user as the user's skills, preferences, and abilities become known and change.

3.1.1 User Modelling Dimensions

Some systems store individual information only for a single characteristic, others model users along multiple dimensions (Figure 5), which may include the following [sosnov2007]:

- Users' knowledge: relies on a fine-grained conceptual structure of a learning domain; thus the aggregate knowledge model consists of knowledge assessments for particular domain concepts.
- Users' cognitive skills: it represents procedural ("how-to") knowledge comparing to concepts representing declarative ("what-is") knowledge. It is used in intelligent tutoring systems.
- Users' background: it is defined as relevant experience gained outside the system before the user started working with it. Unlike knowledge models, a background model is usually static and coarse-grained.

- Concept of a task or need: is a very popular approach employed by different adaptive systems on the Web. Thus, for adaptive recommender systems argue that recommendations not taking into account user information task/need are likely to be meaningless and useless.
- Users' demographic characteristics: information, from the very basic, like gender, age or native language to more complex socio-cultural parameters, such as level of formal education and family income are important features as well. Adaptation to demographic information is widely used in adaptive e-commerce systems and personalized ubiquitous applications as well as in educational environments.
- Users' context: it can include any information about the user's location, time, physical and social environment, the device being used, etc. Currently, the most popular class of applications adapting to the user's context are various kinds of personalized guides and tours.

When talking about user modelling, most of the time an able-bodied user is imagined. This is not always the case. Users with disabilities and elderly people have to be considered as well. Thus, different kinds of impairments (which include colour blindness, low vision, blindness, deafness as well as physical, mental, emotional disabilities and others) have to be taken into account and modelled accordingly.

3.2 User modelling formalisms

Various technologies are used to represent a user model as such. As illustrated in Figure 5, technologies can be arranged according to the level of abstraction. On one end there are pure machine or computer understandable (readable) technologies such as XML (or RDF for that matter). On the other side, there are technologies which are more descriptive and understandable to humans, such as GUMO, ISO/IEC 24751 or UML.

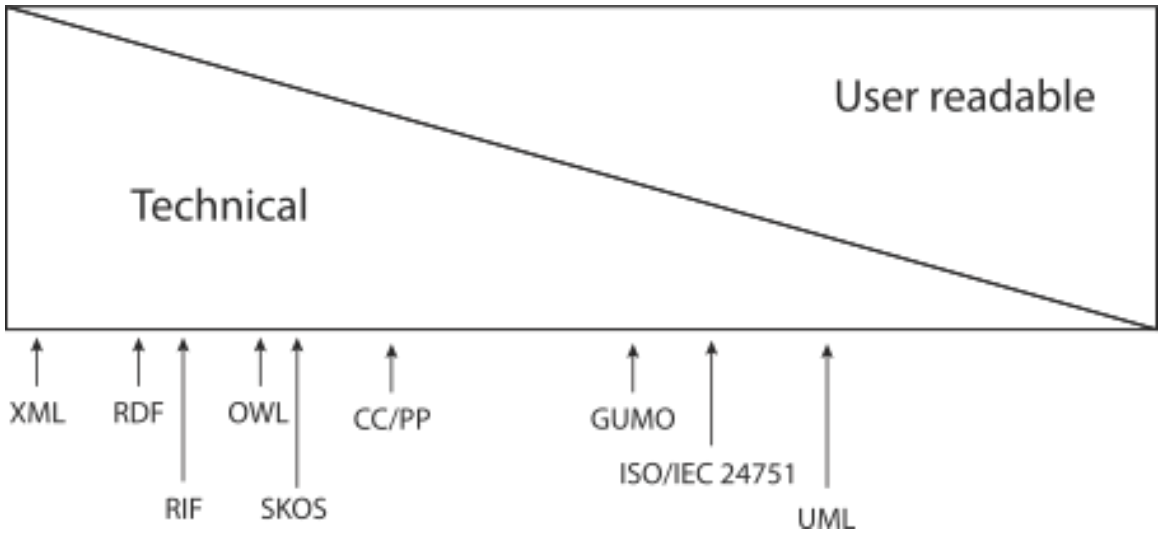

Figure 5: Levels of abstraction in user modelling.

In the field of user modelling a large number of approaches to user model representation is available. Many of them quite specifically serve the purposes of the underlying application. In these cases, specific representation languages with special semantics are used. It is often difficult to relate and compare such approaches to representation mechanisms of other user

modelling systems.

Various representation formalisms are illustrated in Figure 6. At the bottom layer there is XML, which is a general language construct that can be useful for different applications. On top of XML other technologies are found, for example RDF (see section 2.1), RIF and ISO/IEC 24751. RDF is a good starting point for many other formalisms which are used today, not only for user modelling, but other modelling purposes as well (e.g., device modelling, application modelling). Semantic Web technologies like RDF and OWL, help to create other frameworks such as CC/PP (section 2.2) and SKOS. GUMO, which stands on top of OWL, is even more focused on user modelling. There is also UML, which stands on its own (it is not necessarily XML-based, see section 5.2 too). Nevertheless, some UML tools, useful for modelling with UML, offer an option to save data in XML format.

Representation formalisms for user modelling should be powerful enough to satisfy the needs of a range of user modelling systems. A more detailed overview of the technologies in conjunction with user modelling follows in the next chapters.

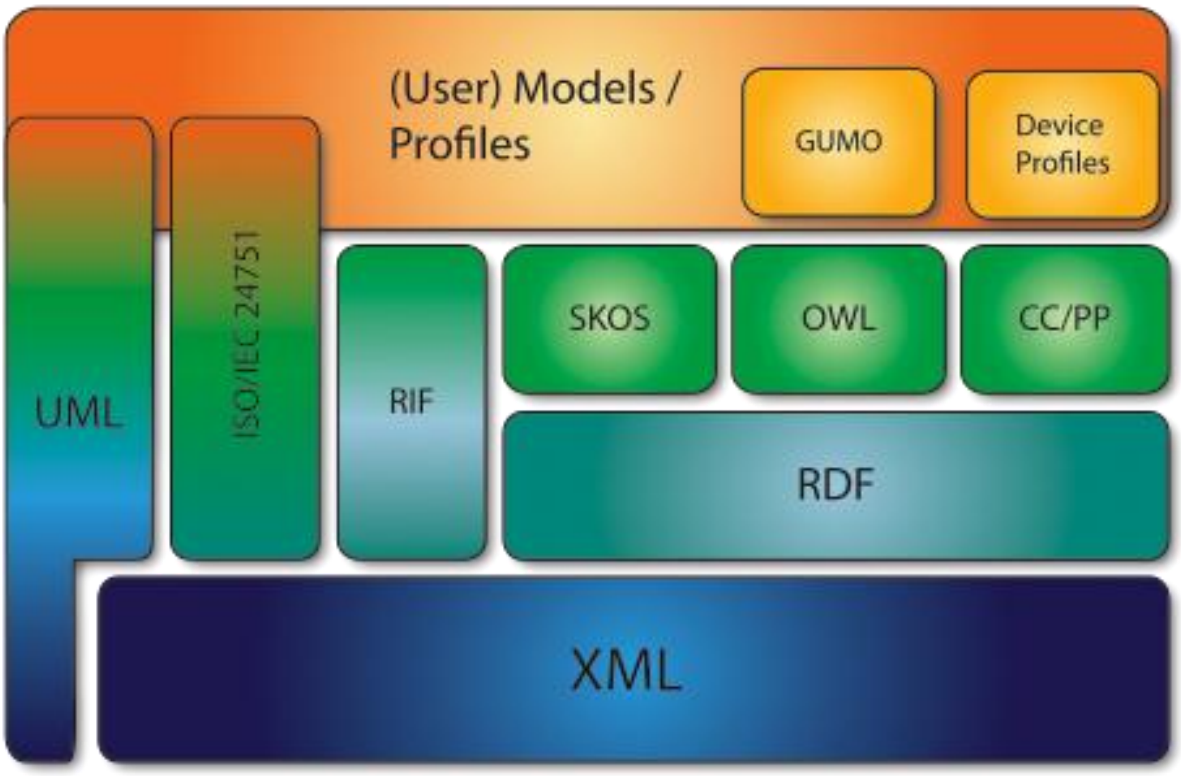

Figure 6: User modelling reference stack.

3.2.1 User modelling frameworks/ontologies

3.2.1.1 Web Ontology Language – OWL

Since ontologies have been developed and investigated in artificial intelligence to facilitate knowledge sharing and reuse, they should form the central point of interest for the task of exchanging user models. XML is designed to serve as an interchange format for weakly structured data. However, XML is purely syntactic and structural in nature. The RDF standard has been proposed as a data model for representing metadata. Nonetheless, the web ontology language OWL has more facilities for expressing semantics along with a greater

machine interpretability than XML and RDF. It adds more vocabulary for describing properties and classes.

OWL is intended to be used when information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms [webOWL01].

There are three sublanguages to OWL, which are designed for use by specific communities of implementers and users. These are:

- OWL Lite: supports users primarily in need of a classification hierarchy and simple constraints.
- OWL DL (Description Logics): supports users who want maximum expressiveness while retaining computational completeness and decidability. OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class).
- OWL Full: supports users who want maximum expressiveness and syntactic freedom of RDF with no computational guarantees.

There are some limitations to OWL, namely:

- Relationships are directed.
- There is no direct language support for n-ary relationships. For example, modellers may wish to describe the qualities of a relation, to relate more than 2 individuals, or to relate an individual to a list. This cannot be done within OWL. They may need to adopt a pattern which encodes the meaning outside the formal semantics instead.

3.2.1.2 Simple Knowledge Organization System – SKOS

SKOS is a common data model for knowledge organization systems such as thesauri, classification schemes, subject heading systems and taxonomies. Using SKOS, a knowledge organization system can be expressed as machine-readable data. It can then be exchanged between computer applications and published in a machine-readable format in the Web.

The SKOS data model is formally defined in as an OWL Full ontology. It is based on a concept-centric view of the vocabulary, where primitive objects are not terms, but abstract notions represented by terms. Each SKOS concept is defined as an RDF resource. Each concept can have RDF properties attached, including:

- one or more preferred index terms,
- alternative terms or synonyms,
- definitions and notes along with specification of their language.

SKOS data are expressed as RDF triples, and may be encoded using any concrete RDF syntax (such as RDF/XML or Turtle). The SKOS data model provides support for four basic types of mapping, which is intended to provide a vocabulary to express matching of concepts from one concept scheme to another. Different types of mapping link include: hierarchical, associative, close equivalent and exact equivalent. To declare relationships between

concepts with more specific semantics than the simple "broader-narrower", such as class-instance or partitive relationships SKOS extensions are used [webSkos].

3.2.1.3 Rule Interchange Format – RIF

In contrast to other semantic Web standards such as RDF and OWL, RIF is focused on exchange rather than development of a single one-fits-all rule language. RIF includes three dialects:

- Core dialect: it comprises a common subset of most rule dialect. RIF Core is a subset of both RIF Basic Logic Dialect (RIF BLD) and RIF Production Rule Dialect (RIF PRD).
- Basic Logic Dialect: adds features to the Core dialect that are not directly available such as: logic functions, equality in the then-part and named arguments. RIF BLD corresponds to logic programs without functions or negations. RIF BLD has a model-theoretic semantics.
- Production Rule Dialect: it specifies an abstract syntax that shares features with concrete production rule languages (Figure 7), and it associates the abstract constructs with normative semantics and a normative XML concrete syntax (Figure 8).

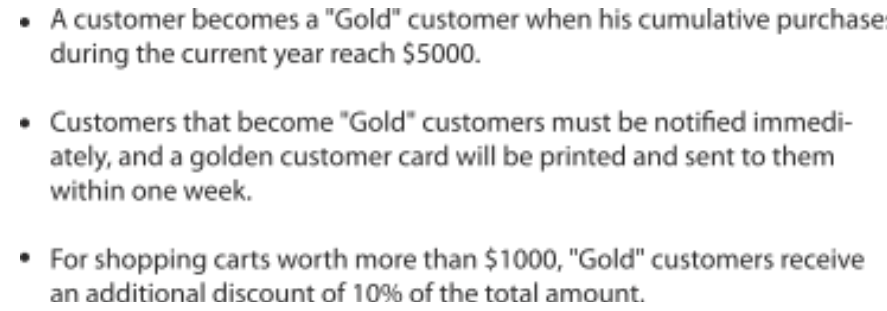

Figure 7: Examples of production rules [webRif].

```
Prefix(ex <http://example.com/2008/prd1#>)
(* ex:rule_1 *)
Forall ?customer ?purchasesYTD (
 If   And( ?customer#ex:Customer
           ?customer[ex:purchasesYTD->?purchasesYTD]
           External(pred:numeric-greater-than(?purchasesYTD 5000)) )
 Then Do( Modify(?customer[ex:status->"Gold"]) ) )
```

Figure 8: Presentation of the first rule from Figure 7 [webRif].

The family of RIF dialects is intended to be uniform and extensible. That means that dialects are expected to share as much as possible of the existing syntactic and semantic apparatus. Extensibility here means that it should be possible for motivated experts to define a new RIF dialect as a syntactic extension to an existing RIF dialect, with new elements corresponding to desired additional functionality [webRif].

3.2.2 Top-level models and ontologies

Many ontologies can be used for user modelling. Nevertheless, a more specific and commonly accepted top-level ontology could be of great importance for the user modelling research community. Such ontology should be represented by one of the underlying semantic web languages like OWL and thus be available for all user adaptive systems at the

same time. A major advantage would be the simplification for exchanging user model data between different user-adaptive systems.

3.2.2.1 General User Model Ontology – GUMO

As already mentioned, the benefit of a top level ontology would be a simple exchange of user models and their data among different systems. GUMO is one of those ontologies. GUMO overcomes the problem of syntactical and structural differences among existing user modeling systems. It is influenced by UserML, SUMO and UbisWorld and its main conceptual idea is the division of user model dimensions into the three parts: auxiliary, predicate and range.

In GUMO, design is based on USERML approach, which means that approximately 1000 groups of auxiliaries, predicates and ranges have so far been identified and inserted into the ontology. Identified user model auxiliaries are *hasKnowledge, hasInterest, hasBelieve, hasPlan, hasProperty, hasGoal, hasPlan, hasRegularity* and *hasLocation.* This listing is not intended to be complete, but it is a start with which a lot of user facts can be realized. For example, if one wants to say something about the user's interest in football, one could divide this so-called user model dimension into the *auxiliary* part *has interest*, the *predicate* part *football* and the *range* part *low-medium-high*. It turned out though, that actually everything can be a predicate for the auxiliary hasInterest or hasKnowledge, which leads to a problem if work is not modularized. The solution GUMO used is to identify basic user model dimensions on the one hand while leaving the more general world knowledge open for already existing other ontologies on the other. This means that it uses a modular approach which is one of the key features of GUMO.

```
<owl:Class rdf:ID="PhysiologicalState.700016">
  <rdfs:label> Physiological State </rdfs:label>
  <rdfs:subClassOf rdf:resource="#BasicUserDimensions.700002" />
  <gumo:identifier> 700016 </gumo:identifier>
  <gumo:lexicon>state of body or bodily functions</gumo:lexicon>
  <gumo:privacy> high.640033 </gumo:privacy>
  <gumo:website rdf:resource="&GUMO;concept=700016" />
 </owl:Class>
```

Figure 9: Sample of GUMO code [heck2005, heck2006].

Another important feature of GUMO is the ability of multiple-inheritance. For example, happiness is defined as rdf:type of the class EmotionalState and FiveBasicEmotions. Thus GUMO allows constructing complex, graph-like hierarchies of user model concepts, which is especially important for ontology integration [heck2005, heck2006].

3.2.2.2 ISO/IEC 24751

ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form a specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of international standards through technical committees established by the respective organization dealing with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest.

ISO/IEC 24751 is derived from the IMS GLC Learner Information Package Accessibility for LIP Specification and the IMS AccessForAll Meta-data Specification. It is intended to address mismatches between personal needs and preferences caused by any number of circumstances, including requirements related to client devices, environments, language proficiency or abilities. The purpose is not to point out flaws in educational digital resources with respect to accessibility and adaptability, but to facilitate the discovery and use of the most appropriate content components for each user.

It provides a common framework to describe and specify learner needs and preferences on the one hand and the corresponding description of the digital learning resources on the other hand so that individual learner preferences and needs can be matched with the appropriate user interface tools and digital learning resources.

In the first part of the standard a common framework for additional parts is provided. These additional parts provide two complementary sets of information [iso24751p1]:

1. the description of a learner's accessibility needs and preferences, including
    - how digital resources are to be displayed and structured,
    - how digital resources are to be controlled and operated, and
    - what supplementary or alternative digital resources are to be supplied;
2. the description of the characteristics of the resource that affect how it can be perceived, understood or interacted with by a user, including:
    - what sensory modalities are used in the resource,
    - the ways in which the resource is adaptable (i.e., whether text can be transformed automatically),
    - the methods of input the resource accepts, and
    - the available alternatives.

At the same time, it currently does not support the following:

- individual needs and preferences and resource descriptions related to non-digital learning resources and adaptability of learning resources in non-computer-mediated environments;
- individual needs and preferences and digital resource descriptions related to digital resource delivery with respect to technical requirements of a user's device;
- individual needs and preferences and digital resource descriptions related to culture and language;
- individual needs and preferences and digital resource descriptions related to location-based services;
- individual needs and preferences and digital resource descriptions related to events and places;
- individual needs and preferences and digital resource descriptions related to time-based services;
- individual needs and preferences and digital resource descriptions related to multi-granularity of aggregations of heterogeneous resources and services.

The second part of the standard provides a common information model for describing the learner or user needs and preferences when accessing digitally delivered resources or services. This description is one side of a pair of descriptions used in matching user needs and preferences with digital delivery. This model divides the personal needs and preferences of the learner or user into three categories [iso24751p2]:

- Display: how resources are to be presented and structured;
- Control: how resources are to be controlled and operated; and,
- Content: what supplementary or alternative resources are to be supplied.

The third part of this standard meets the needs of learners with disabilities and of anyone in a disabling context and it provides a common language to describe digital learning resources to facilitate matching of those resources to learners' accessibility needs and preferences. It provides a machine-readable method of stating user needs and preferences with respect to digitally based education or learning [iso24751p3].

It focuses on the description of the characteristics of the resource that affect how it can be perceived, understood or interacted with by users, including:

- what sensory modalities are used in the resource,
- ways in which the resource is adaptable, i.e., whether text can be transformed automatically,
- which methods of input the resource accepts, and
- which adaptations are available.

Information model for describing learning resources is provided thus individual learner preferences and needs can be matched with the appropriate user interfaces, tools and learning resources within a computer-mediated learning environment.

3.2.2.3 Unified Modelling Language – UML

UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. UML includes a set of graphic notation techniques to create visual models. A more general overview of the language is given in section 5.2. Here, the focus is on class diagrams of UML, which are found very useful in the user modelling process.

The class diagram is the main building block in object oriented modelling. It is used both for a general conceptual modelling of the application's systematics, and for a detailed translation of the models into programming code. The classes in a class diagram represent main objects and/or interactions in the application, and also the objects to be programmed. In the class diagram these classes are represented with boxes (Figure 10).

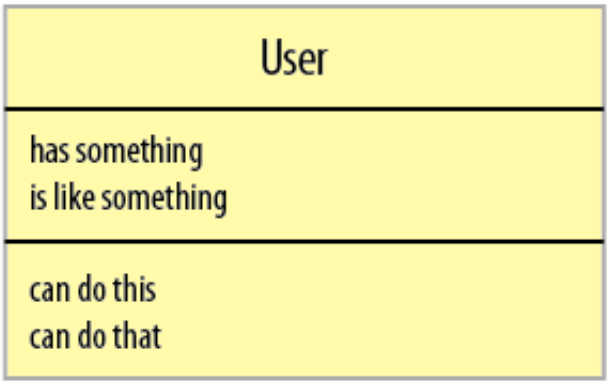

Figure 10: Sample box representing a User class.

The upper part of the box representing the class holds the name of the class, the middle part contains the attributes and the bottom part gives the methods or operations the class can take or undertake. Different mechanisms are provided to represent class members such as attributes and methods and additional information about them. Among classes, different relationships can be defined. These are specific types of logical connections found in class and object diagrams. Types of relationships can be the following.

- Links: link is represented as a line connecting two or more object boxes, it is an instance of an association – it creates a relationship between two classes.
- Association: it represents a family of links. An association can be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility, and other properties. There are five different types of association. Bi-directional and uni-directional associations are the most common ones. Association represents the static relationship shared among the objects of two classes.
- Aggregation: aggregation is more specific than association. As a type of association, an aggregation can be named and have the same adornments that an association can. However, an aggregation may not involve more than two classes. Aggregation can occur when a class is a collection or container of other classes, but where the contained classes do not have a strong life cycle dependency on the container—essentially, if the container is destroyed, its contents are not. Aggregation is shown in Figure 11.

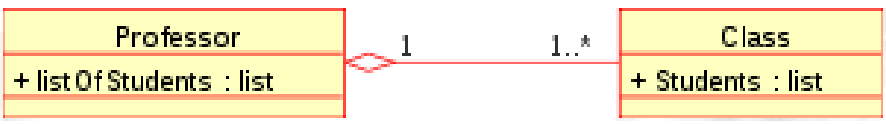

Figure 11: Aggregation between two classes - represented as hollow diamond shape. [webUML]

- Composition: composition is more specific than aggregation. Composition usually has a strong life cycle dependency between instances of the container class and instances of the contained class(es): If the container is destroyed, normally every instance that it contains is destroyed as well. Graphical representation of a composition relationship is a filled diamond shape.
- Generalization: it indicates that one of the two related classes (the subclass) is

considered to be a specialized form of the other (the super type) and superclass is considered as 'Generalization' of subclass. This means that any instance of the subtype is also an instance of the superclass. The generalization relationship is also known as the inheritance.

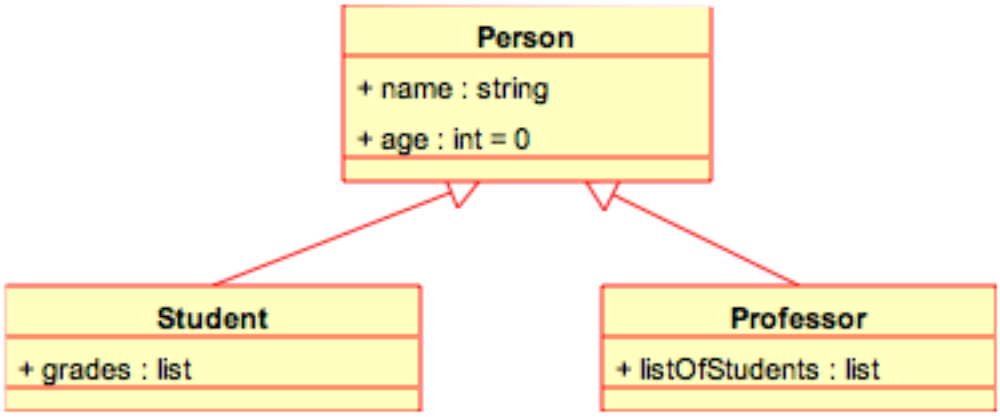

Figure 12: Generalization between one superclass and two subclasses.

- Realization: is a relationship between two model elements in which one model element (the client) realizes (implements or executes) the behaviour that the other model element (the supplier) specifies.
- Dependency: is a weaker form of relationship which indicates that one class depends on another because it uses it at some point of time. Dependency exists if a class is a parameter variable or local variable of a method of another class.

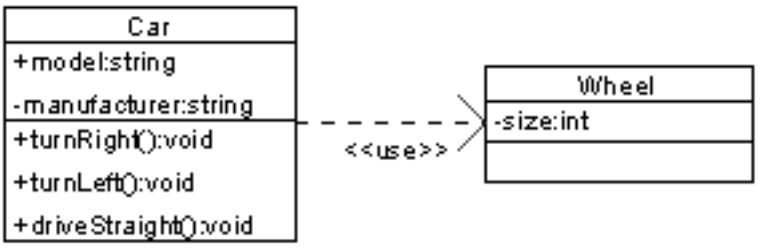

Figure 13: Dependency between Car class and Wheel class.

To associate user model classes with other diagrams, or with the system, special classes can be used. One of those is the boundary class. Boundary classes handle communication between actors and the system's internal components. They can be user interfaces, system interfaces or device interfaces. Entity classes can also be used. They model the information handled by the system, and sometimes the behaviour associated with the information. Control classes are available as well. They are used to handle the flow of control for a use-case and can therefore be seen as coordinating representation classes.

Beside class diagrams, UML offers use case diagrams. These are a type of behavioural diagram used to present a graphical overview of the functionality provided by the system. The diagram shows which system functions are performed for which actor (user). They are used mainly for application modelling, which is discussed in this document as well. Hence, a link between application modelling and user modelling is obvious.

A similar method for modelling user is already in use in ISO/IEEE 11073-10201 – Health

informatics point-of-care medical device communication. They use a simplified version of class diagrams to show the so-called Patient Package Model, which deals with all patient-related information relevant in the scope of the standard. The model is shown in Figure 14 [webIeeeIso].

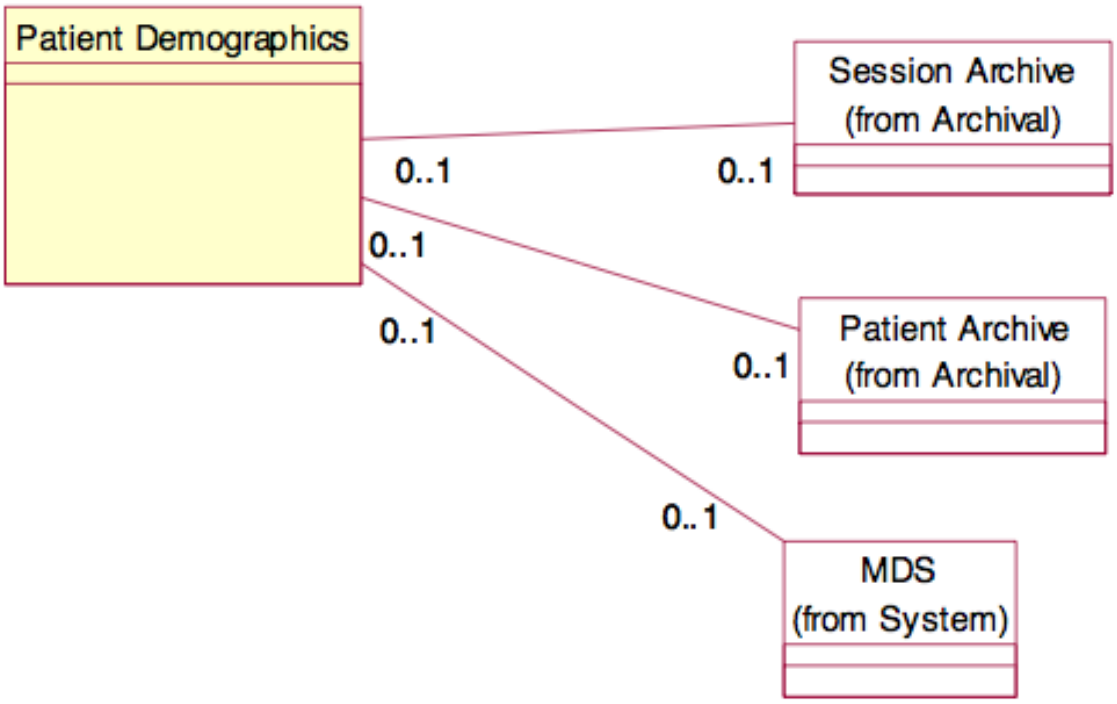

Figure 14: Patient Package model [webIeeeIso].

3.3 User Model for I2Web

Even users that could be classified as able-bodied in a carefully controlled environment such as an office, can become environmentally impaired once they move to areas that are noisy or subject to perturbations such as vibration. As the user base and environmental circumstances change, existing user model will become increasingly strained and user modelling techniques and interface design methods have to change accordingly.

For user models to be utilized in achieving universal access, it is most logical to begin with a simple model. It is necessary to validate any user model before assuming that the theoretical basis behind it is correct for a particular user group [keates2001]. This approach is the one to be evaluated under the scope of the project.

4 Device modelling

For the I2Web project, the concept of device is quite broad. We are in an age of ubiquitous access to Web 2.0 applications where desktop PCs, smart phones, tablets or TVs could be used. Thus our analysis of the state of the art is focused on models that are interoperable with user and application models, thus we can build our adaptation process on semantic web technologies.

It can be noted that most of the frameworks come from the Mobile Web arena, where the first serious attempts to model mobile telephones started at the beginning of the previous decade.

4.1 User Agent Profile (UAProf) specification

UAProf is based on CC/PP and has a base of existing implementations for many devices. Lists of hundreds of devices can be found at the UAProf profile repository of w3development.de,[2] in the DELI repository[3] and in the OMA site.[4] In the description below, UAProf is demonstrated as a good candidate for use in I2Web.

The Open Mobile Alliance developed this standard to allow the communication of the capabilities of mobile devices throughout the Internet in order to receive the content adapted to the characteristics of the users mobile device. Version 2 of the specification is well known as UAProf or OMA User Agent Profile [uaprof20]. The design was guided by the desire to solve one of the main problems of the development of web applications for multi-device environments: how to know the characteristics of the device that is going to present/display the content. UAProf is the first implementation of the CC/PP framework. Many mobile manufacturers published in their web site the RDF files that describe the functionality of their devices. However, this is disappearing with the arrival of the smart phones, which capabilities are many times equivalent to desktop computers..

4.1.1 Operation

The user agent profile is concerned with capturing classes of device preference information. These classes include (but are not restricted to) the hardware and software characteristics of the device as well as information about the network to which the device is connected.

As a request travels over the network from the client device to the origin server, each network element may optionally add additional profile information to the transmitted User Agent Profile. These additions may provide information available solely to that particular network element. Alternatively, this information may override the capabilities exposed by the client, particularly in cases where that network element is capable of performing in-band content transformations to meet the capability requirements of the requesting client device. This overriding mechanism is ideal when used in combination with user preferences.

[2] http://w3development.de/rdf/uaprof_repository/
[3] http://delicon.sourceforge.net/profiles.html
[4] http://validator.openmobilealliance.org/VALIDATED/

The information is prepared on the client device, optionally enhanced with information provided with a particular request, optionally combined with other information available over the network, and made available to the origin server. Over the Internet, this specification assumes the use of the CC/PP, HTTP 1.1, and optionally the CC/PP Exchange Protocol over HTTP, and HTTP 1.1 with the HTTP Extension Framework. The protocol used to retrieve information stored in the profile repository is not specified in this specification. Figure 15 represents this process:[5]

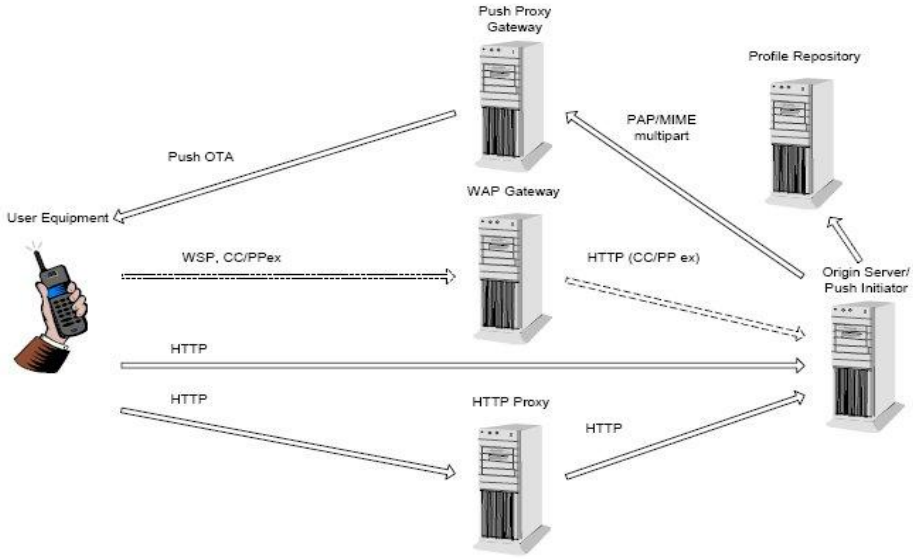

Figure 15. UAProf end-to-end architecture.

4.1.2 UAProf schema

The schema for User Agent Profiles consists of description blocks for the following key components:

- **HardwarePlatform**: A collection of properties that adequately describe the hardware characteristics of the terminal device. This includes the type of device, model number, display size, input and output methods, etc.
- **SoftwarePlatform**: A collection of attributes associated with the operating environment of the device. Attributes provide information on the operating system software, video and audio encoders supported by the device, and user's preference on language.
- **BrowserUA**: A set of attributes to describe the HTML browser application.
- **NetworkCharacteristics**: Information about the network-related infrastructure and environment such as bearer information. These attributes can influence the resulting content, due to the variation in capabilities and characteristics of various network infrastructures in terms of bandwidth and device accessibility.
- **WapCharacteristics**: A set of attributes pertaining to WAP capabilities supported on the device. This includes details on the capabilities and characteristics related to the WML Browser, etc.
- **PushCharacteristics**: A set of attributes pertaining to Push specific capabilities supported by the device. This includes details on supported MIME types, the

[5] See UAProf 2.0 specification

maximum size of a push-message shipped to the device, the number of possibly buffered push-messages on the device, etc.

Additional components can be added to the schema to describe capabilities pertaining to other user agents such as an email application or hardware extensions.

UAProf makes use of namespaces that can be used to construct Profile documents. They are defined and fixed by the specification. These namespaces are defined as:

Namespace	URI	Prefix
Resource Description Framework (RDF)	http://www.w3.org/1999/02/22-rdf-syntax-ns#	rdf
Composite Capability/ Preferences Profiles	http://www.wapforum.org/profiles/UAPROF/ccppschema-20010430#	prf

The specification builds upon and coexists with numerous Internet standards. These relationships are summarized below:

- Composite Capability/Preferences Profiles (CC/PP): This specification defines the information structure according to the structure mandated by the CC/PP specification note (see section 2.2).
- Resource Description Framework (RDF): The CC/PP specification uses the RDF syntax to represent the information. In designing or extending the schema, a schema designer must be familiar with the RDF concepts (see section 2.1).
- Wireless Session Protocol (WSP): information is transmitted over wireless networks within WSP headers.
- WAP Binary XML (WBXML): When the information is transmitted over the WSP protocol, this specification mandates that it be encoded according to the WBXML specification.
- CC/PP Exchange Protocol over HTTP: When transmitted over the Internet, this specification requires that the information be transmitted using the CC/PP Exchange Protocol over HTTP which, in turn, defines headers in HTTP 1.1 with the HTTP Extension Framework.
- WAP Push Access Protocol (PAP): This protocol is used by push origin servers to retrieve the information from the WAP gateway or Push Protocol Proxy (PPG). The request is issued over HTTP, and the response contains the profile, with MIME type text/xml.

4.1.3 Problems and open issues

After several years of life, many organizations think that this specification does not improve the complex world of the development for multi-device environments. There is no homogeneity between the manufacturers in the use of CC/PP rules and so also in use of UAProf.

Furthermore, not all UAProf profiles that do exist are reliable. The indicated values are not always necessarily accurate and sometimes require an exhaustive process of testing to

correct these errors. UAProf is not a trustworthy source of device information, because a wrong URL can be defined by the mobile device, or the UAProf server could be not available.

Finally, the information provided by the UAProf profiles may be incomplete for the objective that was designed. In order to provide an optimal adaptation of the content on any device there is key information that it is not gathered in UAProf, among this we could list:

- CSS version supported.
- No support for CSS attributes.
- Size of the scroll bars.
- Dimensions of the navigation area.
- Maximum size to download multimedia content.
- etc.

4.2 Other standards relating to device modelling

4.2.1 WURFL

Wireless Universal Resource File (WURFL[6]) is a configuration file that is intended to contain information about wireless devices, with an aim to provide device information to content servers, primarily through the web browser. WURFL is an XML configuration file which contains information about capabilities and features of many mobile devices. WURFL is basically an extension of UAProf, with the main difference is that WURFL is not designed to be dependent on servers supplying missing device information in the http header.

This project is open-source and is intended for developers working with the WAP and wireless. The WURFL site lists the following differences between it and UAProf:

- UAProf relies on an infrastructure to request profiles.
- There is no guarantee that the info in UAProf are accurate, WURFL provides the programmer a chance to fix errors on the fly.
- WURFL allows modelling any feature or capability of whatever phone, no matter what the phone manufacturer does or say. You are not limited to the properties in UAProf.
- The WURFL can be installed at any site and does not need to grab device profiles off a repository on the net.
- WURFL is about developer-centred rather than manufacturer centred as UAProf.

WURFLs main disadvantage for I2Web is its lack of integration with existing web standards, in particular with Semantic Web technologies.

4.2.2 URC and V2

There is another approach to device independence, which is to adapt the device to the content, rather than the content to the device. The chief system that incorporates that approach is the URC (Universal Remote Control) standard, which migrated into the V2 standard. The difference between the content personalization approach and URC approach is that the content personalization supplies appropriate content to fit the devices

[6] http://wurfl.sourceforge.net/

capabilities, whereas the URC approach is to modify the GUI presented to the user to reflect the control requirements of the target device. Later in its life, the URC Consortium[7] was absorbed by the technical committee at the International Committee for Information Technology Standards (INCITS) that is charged with developing national standards for Information Technology Access Interfaces and became the V2 project.[8]

4.2.2.1 History and purpose of the URC standard

The Universal Remote Console (URC) framework is a set of ANSI/INCITS standards (an ISO version has been produced). They define a generic framework and an XML-based user interface language to use any device to act as a remote control in order to monitor or control electronic devices called “targets” [zimmermann2004].

The ISO standard is being developed by the Working Group 8 of ISO/IEC JTC1 SC35 “User Interfaces”. This ISO multipart standard is composed of five documents:

- ISO 24752-1: Information technology - User interfaces - Universal remote console - Part 1: Framework.
- ISO 24752-2: Information technology - User interfaces - Universal remote console - Part 2: User Interface Socket Description.
- ISO 24752-3: Information technology - User interfaces - Universal remote console - Part 3: Presentation Template.
- ISO 24752-4: Information technology - User interfaces - Universal remote console - Part 4: Target Description.
- ISO 24752-5: Information technology - User interfaces - Universal remote console - Part 5: Resource Description.

4.2.2.2 Main components within the URC framework

The standard specifies communications between a Target (a resource which a user wants to control), and a Universal Remote Console (URC) that provides the user with a user interface through which the Target can be controlled. The URC software is typically hosted on the user's terminal, however the standard also considers the possibility to have a distributed approach. Communications between the Target and URC take place over a network, the so-called Target-URC Network. The network protocol to be used is not specified by the standard.

The protocols defined by the standard are used to provide discovery of Targets, and to establish and maintain control sessions between URCs and Targets:

- Targets and URCs access the Target-URC Network through Target-URC Network Links.
- Targets support discovery by providing essential information in a Target Description.
- Each Target provides a User Interface Socket (or short "Socket"), or set of Sockets, through which a URC can access part or all of the Target's internal states and provide control inputs to the Target.

[7] http://myurc.org/
[8] http://www.ncits.org/tc_home/v2.htm

- For each Socket, a Target provides a User Interface Socket Description (also known as "Socket Description") which describes the Socket in a machine readable and interpretable manner.
- Additionally, the Target provides Resources that pertain to the user interface of the Target, as viewed through that Socket.
- The Socket Description and Resources are used by the URC to find or generate an appropriate user interface, given the functionality of the Target, the nature of the URC device, and the user's interaction preferences.

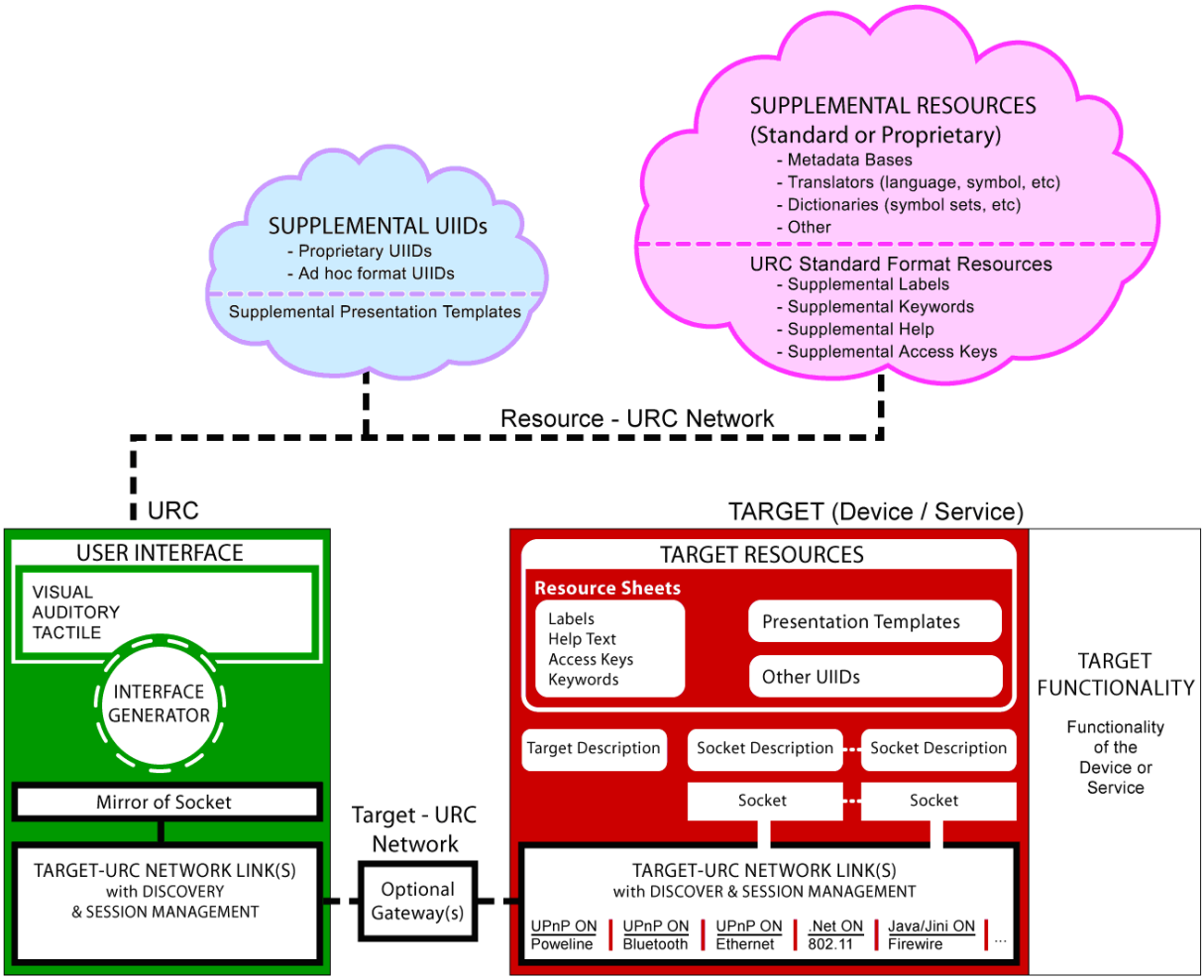

Figure 16. URC Architecture Diagram.

The phases in the interaction between a Target and a URC are the following (see Figure 16):

- The discovery phase initializes the URC to locate and identify all available Targets and their Sockets.
- During the control phase, a session is initiated, maintained and terminated between Target and a URC. The purpose is controlling a functional unit of the Target via the Socket.

The User Interface Implementation Description (UIID) is a type of resource:

- A UIID is a description of a user interface for a particular Socket.

- The UIIDs provide a mechanism by which a manufacturer can provide tuned interfaces for their Target Sockets which are predefined and optimized to work on particular URCs.
- The UIIDs can be provided by a Target or by external sources.
- The Presentation Template constitutes one general form of UIID.

An Atomic Resource is another resource: it is an object within a user interface that is used as an atomic entity in the construction of a concrete user interface:

- Atomic Resources include labels, help text, access keys and keywords.
- An Atomic Resource may be of any form and modality, including text, images, sounds, animations and video clips.

In order to render their content to a user, Atomic Resources are provided (by the Target and other entities which are external to it) in the form of Resource Sheets. A Resource Sheet is a file that contains descriptions of Atomic Resources (or "Atomic Resource Descriptions"). A Grouping is a Resource that defines a hierarchical structure for the elements of a Socket or UIID:

- Groupings are useful for constructing the layout and navigation mechanisms of a particular user interface.
- They are provided in Resource Sheets.

A URC may take advantage of Resources from the Target or from a Resource Service. These are referred to as Target Resources and Supplemental Resources respectively:

- Supplemental Resources can be used by a URC to replace, to supplement, or to help interpret or translate, any Resources provided by a Target.
- In general, Resources for building user interfaces may be obtained from the Target, stored on the URC, or gathered from the Internet.
- A URC uses the Target-URC Network to retrieve Target Resources.
- It may use any form of networking or other mechanism to access Supplemental Resources via a Resource-URC Network.
- Mechanisms by which a URC accesses Resources external to the Target are not specified by the standard.

As described above, the URC software that is typically hosted on the user agent (the terminal used to control the target), but the standard also considers the possibility to have a distributed approach. In this line, the Universal Control Hub (UCH) architecture is a specific configuration of the URC framework, using a gateway approach between controllers and targets [zimmermann2006]. This approach enables URC to provide a suitable framework compatible with 'de facto' standards and architectures of the Digital Home market, such as UPnP or similar.

The URC framework has been initially designed for remote control of devices and services where the controller and the target are tightly synchronized. In contrast, Web 2.0 applications have been typically offered by web based information systems in an asynchronous way, thus the applicability of URC to I2Web is very limited.

While the premise of this approach (modifying the user agent to reflect the control needs of the target device) is very much different from the vision of the I2Web system (modifying the content to reflect the needs of the user and abilities of the device), the details of the V2

approach may prove at least inspiration to the design of the more specialised service delivery.

4.2.3 Other device modelling systems

The following is a survey of existing but less applicable device modelling systems.

4.2.3.1 Personal Digital Assistant Profile

There are several other schemas and frameworks for device modelling. Among these is the Personal Digital Assistant Profile (PDAP) [sun2002a], produced by the Personal Digital Assistant Profile Expert Group, defines the architecture and associated APIs for application development for Personal Digital Assistants (PDAs). PDAP provides support for configuring PDAs to adapt to a user's individual needs, like controlling cursor appearance and screen colours. PDAP is based on the Mobile Information Devices Profile (MIDP) 1.0 [sun2002b], which supported specification of target device similarities, to support applications conforming to a PDAs capability.

4.2.3.2 The W3C Device Description Working Group

The objective of the Mobile Web Initiative[9] is to enable access to the Web from mobile devices. It is envisaged that this will typically require adaptation of Web content, which relies on device knowledge. The Device Description Working Group (DDWG[10]) was given the task of defining the means by which such knowledge would be made available to adaptation technology. The group was closed in 2008. Its list is of publications is available here: http://www.w3.org/2005/MWI/DDWG/#publications.

4.2.3.3 Open Mobile Alliance Device Profile Requirements

The Open Mobile Alliance Device Profile Requirements document[11] describes the requirements for managing the publication of a device's hardware, software, and network properties as they change over the course of a data session. Information about these Dynamic Device Properties is used by Application Service Providers to tailor content appropriately for the target device. OMA's User Agent Profile is only capable of conveying information about static device properties.

4.3 Comparisons of the model systems

Below is a table that illustrates some of the differences between some of the systems described above. Special note should be taken of the difference between V2 and CC/PP. V2 is designed to change the appearance and function of the device to reflect control surfaces of devices, CC/PP is designed to support appropriate content to be selected and supplied to the devices. While these two are, in some way, isomorphs of each other; the difference between the approaches illustrates some of the issues involved in device independence / device modelling.

[9] http://www.w3.org/Mobile/
[10] http://www.w3.org/2005/MWI/DDWG/
[11] http://www.openmobilealliance.org/Technical/DM.aspx

System	Domain	Implementation	Notes
CC/PP	Mobile devices	UAProf	RDF based
URC/V2	Remote controls	Some Commercial Demonstrations	Changing device function rather than delivering device appropriate content
PDAP	Small Computers	PDA	Java
UAProf	Mobile Devices	Repositories	RDF

Table 2. Comparison of different device modelling standards.

5 Application models

This section describes the notion of Application Model, and presents some approaches to the problem of modelling applications. The notion of *Application Model* is multifaceted. There area in fact various aspects of an application that can be modelled, and for each of them there are several formalisms/techniques to define and describe models.

Broadly speaking, we can identify at least three major aspects requiring modelling:

- *architecture* design: any complex application should be based on an architectural model that is shared and understood by its designers and developers;
- *implementation* design: when implementing an application it is a good practice to identify known and recurring problems, and to solve them by reusing appropriate and well-established design solutions;
- *interaction* design: an application that interacts with users should carefully define its interaction patterns to improve and simplify usability; this area should take into account user preferences and characteristics (section 3), and (possibly) device characteristics (section 4).

We can therefore say that a full application model is actually a collection of models that give a (possibly formal) representation of the application's interaction patters, architectural choices, and implementation design. Given the broad scope of the notion of Application Model, we focus on some formalisms addressing the three major aspects outlined above.

5.1 Application Model formalisms

In this section we summarize some state of the art formalisms for the definition of application models at the architectural, implementation and interaction levels.

5.2 Architectural models: MDA and UML

OMG Model Driven Architecture (MDA) [mda] envisions a software development process in which application's functionalities and behaviour are specified in a *Platform Independent Model* defined in an abstract modelling language. This model remains stable during the entire application's life-cycle, and it is independent of specific technologies used at the implementation level. The primary intention of this approach is to maximize the return of investment for the organization/company designing and building the application: the Platform Independent Model serves in fact as a stable blueprint architectural reference for implementation and maintenance. OMG advocates the use of (possibly automated) development tools that can translate the Platform Independent Model into a *Platform Specific Model*, and then into a working implementation into any possible platform (Web Services, CORBA, Java, etc.).

Essentially, a Model Driven Architecture usually comprises three macro-levels:

- a top level, consisting of the Platform Independent Model, which gives a platform-neutral representation of the business process and roles of the application
- an intermediate level, consisting of the Platform Specific Model, which specifies every aspect of a coded application as a model (as opposed to code)

- a bottom level, consisting of one or more implementations comprising code and auxiliary files whose bundle represents some executable form of the application.

Various development tools may support (and possibly automate) the translation of modelling artefacts from the top level down to the bottom level.

The overall MDA is supported by OMG industry standard modelling specifications, including the well known and widely adopted *Unified Modeling Language* (UML) [uml].

UML is a general purpose modelling language, and it is (possibly) the most used formalisms for modelling applications at the architectural level. In general a UML model consists of a set of diagrams and a set of associated documents. Each diagram provides a partial graphic representation of a system's model. Associated documents usually provide additional information (such as use case textual descriptions).

The UML specifications (currently at version 2.4 [uml2011]) essentially defines a “graphical language for visualizing, specifying, constructing, and documenting the artefacts of distributed object systems” [omgcat].

UML 2.0 defines thirteen types of diagrams, that can be divided in three categories [siegel2005]:

- *Structural diagrams*: describe the static structure of the application in terms of objects, attributes, operations and relationships. Structural diagrams include the class diagram, object diagram, component diagram, composite structure diagram, package diagram, and deployment diagram.
- *Behavioural diagrams*: describe the dynamic behaviour of the application components by illustrating collaboration among objects, and changes to their internal state. Behavioural diagrams include the use case diagram (usually adopted during the requirements elicitation phase), activity diagram, and state machine diagram.
- *Interaction diagrams*: also describe behavioural aspects of the system, but focusing on some type of interaction among the different elements in the model. Interaction diagrams (all derived from the more general Behaviour Diagram) include the sequence diagram, communication diagram, timing diagram, and interaction overview diagram.

Rather than describing in details the different UML diagrams, we refer the reader to the ample documentation available on the Web (including [wiki_uml, siegel2005, umldiag, omglinks]) and in the press (including [fowler2004, booch2005, ambler2005, ambler2004]).

5.3 Implementation models: Design Patterns

At the implementation level it is possible to identify commonly occurring problems. Software engineers usually address such problems by using *design patterns*. A design pattern is essentially a model of a general reusable solution, and, when applied, it is instantiated by implementing it in a specific programming language and technology platform.

Noticeably, traditional design patterns rely on the object-oriented programming paradigm, and may not be directly applicable to other kind of languages (such as functional languages). However, such design pattern are so widely used that they constitute a valuable set of solution models (or templates) for common problems in software engineering.

Historically, Kent Beck and Ward Cunningham were among the firsts to experiment and work on the idea of applying design patterns to programming [beck1987]. There are numerous books on the topic, including [gamma2000, shalloway2000, metsker2006], and ample documentation on the Web (see [wiki_dpcs] for an overview and for further references to other Web sites on the subject).

Traditional design patterns are grouped into three categories [gamma2000]:

- Creational Patterns: these patterns deal with techniques providing fine grained control over objects' and classes' creation (some examples include factories, lazy initialization, object pools, and singleton).
- Structural Patterns: these patterns deal with techniques for composing objects to form larger structures, and for establishing relations among objects (some examples include patterns for adaptation, decoration, and composition).
- Behavioural Patterns: these patterns deal with algorithms implementation and with communication among objects (some examples include observer, strategy, and state patterns).

Additionally, other categories of design patterns have been proposed. Such additional categories include:

- Concurrency patterns: these patterns deal with problems arising in the multi-threading and parallel programming paradigm (some examples include scheduling and thread pooling [mattson2005, schmidt2000].
- Enterprise Application Integration patterns: these patterns deal with problems arising when performing system integration of enterprise-level computer applications [hohpe2006, fowler2003].
- Service-oriented architecture patterns: these patterns deal with problems arising when implementing system functionalities as a set of interoperable and loosely coupled and distributed services [erl2009].
- Architectural patterns: these patterns are a sort of extension of design patterns that usually have a broader perspective, addressing various issues such as performance, high availability, maintenance, and business risk minimization of the overall application architecture (see [avgeriou2005, buschmann2007]). Some examples of architectural patterns are widely used in modern Web applications: the Model-View-Controller (originally invented by Trygve Reenskaug [reenskaug1978]; see also [avraham2001, wiki_mvc, fowler2006]), and the more recent Model-View-Presenter [potel1996, wiki_mvp, fowler2006].

The above list comprises the most widely known categories of design patterns. Although other categories may exist (addressing other specific domains), it is important to notice that the underlying benefit of all design patterns is that of providing *models of reusable solutions* for common problems arising when implementing applications.

5.4 Interaction models: model-based user interface design

Interaction models represent the set of actions occurring between the user and the application. UML provides at least two interaction diagram formats: sequence and collaboration diagrams.

Sequence diagrams usually describe the behaviour of objects by showing the messages that they pass each other. Sequence diagrams lay out the message along a vertical lifeline, which represents the time-ordered history of the interaction.

Collaboration diagrams are essentially a graphical representation of a set of objects that communicate with each other. Collaboration (i.e., communication) is shown by lines between pairs of objects; such lines carry (usually numbered) labels representing the messages being exchanged between the objects.

Although sequence diagrams and collaboration diagrams are designed to show interactions among objects, it is possible to use them to give some representation of the interaction between a user and an application.

Other formalisms have been proposed to expressively represent the interactive applications. An increasingly important research area is the *Model-Based User Interface Design*, which aims at identifying high-level models for the specification and analysis of interactive applications from a semantic oriented level (as opposed to a more traditional implementation and purely syntactic level). The World Wide Web Consortium (W3C) hosted an incubator working group[12] on Model-Based UI design. Such working group ended its activities in 2010, and released an interesting final report [mbui2010]. The use model-based techniques aims at abstracting UI designers from implementation details, allowing them to concentrate on the semantics of user interaction, and on effectively manage the increasing complexity of interactive applications; additionally these techniques provide a consistent model that designers can refer to both during the development and maintenance/evolution phase of the application's user interface [paterno2005].

As reported in [mbui2010, paterno2009] it is possible to distinguish at least two generations of approaches to model-based UI design:

- approaches focusing on deriving abstractions for graphical UIs;
- approaches focusing on expressing the high-level semantics of the interaction.

Recently, there has been a renewed interest in model-based UI interaction design, covering also accessibility-related issues. We refer to [mbui2010] for an extensive state of the art description on the topic. Here, we shortly summarize some interesting formalisms and approaches.

A great deal of work in the area of model-based UI design has been carried out within the FP5 IST project CAMALEON [cameleon]. The CAMALEON Unified Reference Framework [calvary2002, calvary2003, mbui2010] addresses both design and run-time phases of model-based user interfaces supporting multiple targets, and multiple contexts. More precisely, CAMALEON's framework describes and analyses user interfaces from the perspective of the *context of use*, that is a structured information spaces including information about (1) a model of the user, (2) a model of the platform (computing, sensing and interaction devices),

[12] http://www.w3.org/2005/Incubator/model-based-ui/

and (3) the description of the physical environment where the interaction takes place. From this perspective, CAMALEON's classifies user interfaces into four categories:

- multi-target user interfaces, which support multiple types of users, platforms and environments,
- adaptive user interfaces, which are essentially context-aware, and can therefore adapt themselves to context changes,
- adaptable user interfaces, which can be (manually) customized based on set of predefined options, and
- plastic user interfaces, which preserve usability across multiple targets.

We observe that there are also other IST projects working on context aware user interfaces. Examples of such projects include MUSIC[13] (specifically targeting self-adaptive context-aware mobile applications), and Morfeo Serenoa[14] (aiming at the creation of a creation of context-sensitive service front-end for providing context awareness and context adaptation capabilities).

The CAMALEON's reference framework provides also a lifecycle definition for user interfaces development. Such a lifecycle comprises four phases:

1. tasks and concepts definition: this phase defines the logical activities (tasks) that the user should perform to achieve her own tasks by using the application, and the domain objects (concepts) that the activities operate on;
2. abstract user interface definition: this phase defines the UI interaction in an independent way with respect to the platforms, devices and the interaction modalities (graphical, vocal, etc.); the result of this phase is essentially equivalent to the Platform Independent Model of OMG's Model Driven Architecture (see section **5.2**);
3. concrete user interface: this phase instantiates the abstract user interface on concrete platform and device, thus providing a more concrete definition of how the user perceives the UI; the result of this phase is essentially equivalent to the Platform Specific Model of OMG's Model Driven Architecture (see section **5.2**);
4. final user interface: this phase produces the source code (in any programming or mark-up language) that ultimately produces the user interface; note that the rendering of the coded UI may depend on the software environment (virtual machine, browser, etc.), and so the final user interface may be rendered in different ways.

The four phases outlined above are related by an abstraction relationships (going from the final user interface up to the task and concepts) and a reification relationships (going from the task and concepts down to the final user interface). Based on the lifecycle phases, CAMALEON defines also different UI development paths, which are based on transformations along the abstraction and reification relationships outlined above.

Finally, CAMALEON addresses the topic of *user interface description languages*, which serve as tools to formally specifying the artefacts the four lifecycle phases. More precisely, a user

[13] http://www.ist-music.eu/
[14] http://www.serenoa-fp7.eu/

interface description language is a formal language used in the area human-computer interaction studies to describe a user interface independently of any implementation technology [guerrero2009]. We refer to [guerrero2009] for an analysis and comparison of different user interface description languages.

Although CAMALEON does not explicitly focus on Web application and accessibility, we observe that the proposed lifecycle can be used also when designing accessible user interfaces for Web applications. A major consideration when deployment accessible Web application is the proper definition of the concrete user interface, which should conform to the WAI-ARIA specification.

The Accessible Rich Internet Applications [wai-aria2011, aria-primer2010] (WAI-ARIA), released by the Web Accessibility Initiative (WAI)[15] of the W3C, specifies how to make Web applications more accessible by people with disabilities. WAI-ARIA specifically addresses modern Web 2.0 applications with dynamic content, and complex user interfaces based on (a combination of) Ajax, HTML, CSS and Javascript [cooper2007, thiessen2007, thiessen2010]. Such applications use a variety of sophisticated techniques to build and render various widgets such as menus, navigation breadcrumbs, buttons, etc. As a consequence it becomes practically impossible for assistive technologies tools (such as screen readers) to distinguish the semantics of these widgets and to know about their actual functionalities. WAI-ARIA addresses this issue by providing an extension of HTML and XHTML. Such an extension essentially consists of:

- states, and property attributes: these additional (X)HTML attributes provide keyboard focus and state properties that can be tied with platform specific accessibility APIs to simplify the overall application interaction through assistive tools;
- role attribute, which is essentially an annotation providing machine-processable semantic information about the purpose of a mark-up element.

WAI-ARIA provides also an RDF-based ontology of roles values found in Accessibility API sets as well as roles representative of document structure. Such a taxonomy helps user agents or authoring tools in determining what properties a given role supports and to assist with accessibility API mapping of these properties.

In summary, the concrete user interface of an accessible Web application should conform to and use the WAI-ARIA states and roles to enable assistive technologies to gain access to the user interface controls generated within a Web page through a combination of mark-up languages and scripting [raggett2008].

5.5 Application Models for I2Web

The overall architectural models of I2Web will exploit OMG UML to provide appropriate definition of systems and components' models. I2Web deals with accessibility of rich Web 2.0 applications. Consequently, application models for I2Web will primarily consider:

- Model-based user interface design, with a focus on WAI-ARIA specification for the concrete user interface definition.

[15] http://www.w3.org/WAI/

- Architectural Design Patterns such as Model-View-Presenter (MVP) and Model-View-Controller (MVC), with a focus on how such patterns should take into account WAI-ARIA specification.

Since the Model-View-Presenter (or Model-View-Controller) are widely adopted design patterns for modern Web application, an interesting research topic is the analysis of how MVP-based (or MVC-based) applications can effectively incorporate WAI-ARIA techniques to build an accessible user interface (View) while preserving isolation from Presenter/Controller logic and Data Model. Additionally, it is also necessary to investigate how MVP/MVC can incorporate I2Web User and Profile Model to take advantage of such information when building an effective and accessible user interface.

6 Conclusions

We have reviewed in the document the state-of-the-art of the user, device and application models. This exercise has shown a wide variety of approaches for the three domains of interest. Future steps include the comparison of the existing approaches, combined with the results of WP3 and WP7 to begin the initial implementations of this workpackage.

References

[ambler2004] Scott W. Ambler (2004). The Object Primer: Agile Model-Driven Development with UML 2.0. Cambridge University Press. ISBN-13: 978-0521540186

[ambler2005] Scott W. Ambler (2005). The Elements of UML 2.0 Style. Cambridge University Press. ISBN-13: 978-0521616782

[aria-primer2010] Lisa Pappas, Rich Schwerdtfeger, Michael Cooper. An introduction to rich Internet application accessibility challenges and solutions. W3C Working Draft 16 September 2010. http://www.w3.org/TR/2010/WD-wai-aria-primer-20100916/

[avgeriou2005] P. Avgeriou and U. Zdun (2005). Architectural patterns revisited - a pattern language. In Proceedings of the 10th European Conference on Pattern Languages of Programs (EuroPLoP 2005), Irsee, Germany, July 2005.

[avraham2001] Avraham Leff, James T. Rayfield (2001). Web-Application Development Using the Model/View/Controller Design Pattern. Enterprise Distributed Object Computing Conference, IEEE International, p. 0118, Fifth IEEE International Enterprise Distributed Object Computing Conference, 2001.

[beck1987] Kent Beck; Ward Cunningham (1987). Using Pattern Languages for Object-Oriented Program. OOPSLA '87 workshop on Specification and Design for Object-Oriented Programming. OOPSLA '87. http://c2.com/doc/oopsla87.html

[booch2005] Grady Booch, James Rumbaugh, Ivar Jacobson (2005). The Unified Modeling Language User Guide (2nd Edition). Addison-Wesley Professional. ISBN-13: 978-0321267979

[buschmann2007] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal (2007). Pattern-Oriented Software Architecture, Volume 1, A System of Patterns. Wiley. ISBN-13: 978-0471958697

[calvary2002] Calvary, G., Coutaz, J., Bouillon, L., Florins, M., Limbourg, Q., Marucci, L., Paternò, F., Santoro, C., Souchon, N., Thevenin, D., Vanderdonckt, J. (2002). The CAMELEON Reference Framework, Deliverable 1.1, CAMELEON Project

[calvary2003] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. (2003). A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, 3, pp. 289–308.

[cameleon] CAMELEON (Context Aware Modelling for Enabling and Leveraging Effective interactiON) Project (FP5-IST4-2000-30104). http://giove.isti.cnr.it/projects/cameleon.html

[ccpp10] Klyne G., Reynolds F., Woodrow C., Ohto H., Butler M. H. and Tran L. (eds.) (2004). Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0, W3C Recommendation 15 January 2004. World Wide Web Consortium. Available at: http://www.w3.org/TR/CCPP-struct-vocab/

[ccpp20] Kiss, C. (ed.) (2007). Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0, W3C Working Draft 30 April 2007. World Wide Web Consortium. Available at: http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430/

[cooper2007] Michael Cooper (2007). Accessibility of emerging rich web technologies: web 2.0 and the semantic web. In Proceedings of the 2007 international cross-disciplinary

conference on Web accessibility (W4A) (W4A '07). ACM, New York, NY, USA, pp. 93-98. http://doi.acm.org/10.1145/1243441.1243463

[erl2009] Thomas Erl (2009). SOA Design Patterns. Prentice Hall. ISBN-13: 978-0136135166

[fowler2003] Martin Fowler (2003). Patterns of Enterprise Application Architecture. Addison-Wesley Professional. ISBN-13: 978-0321127426

[fowler2004] Martin Fowler (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition). Addison-Wesley Professional. ISBN-13: 978-0321193681.

[fowler2006] Martin Fowler (2006). GUI Architectures. http://www.martinfowler.com/eaaDev/uiArchs.html

[gamma2000] Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides (2000). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional. ISBN-13: 978-0201633610

[gfisch2001] Fischer G. (2001), Center for LifeLong Learning & Design (L3D), Department of Computer Science and Institute of Cognitive Science. User Modeling in Human-Computer Interaction, User Modeling and User-Adapted Interaction, Volume 11, Issue 1-2.

[guerrero2009] Guerrero-García, J., González-Calleros, J.M., Vanderdonckt, J., Muñoz-Arteaga, J. (2009). A Theoretical Survey of User Interface Description Languages: Preliminary Results, Proc. of Joint 4th Latin American Conference on Human-Computer Interaction-7th Latin American Web Congress LA-Web/CLIHC'2009 (Merida, November 9-11, 2009). E. Chavez, E. Furtado, A. Moran (Eds.), IEEE Computer Society Press, Los Alamitos, pp. 36-43.

[heck2005] Heckman D., 15 November 2005. Ubiquitous User Modeling. Available at: http://wwwis.win.tue.nl/ah/thesis/DominiksDiss.pdf

[heck2006] Heckmann D., Schwartz T., Brandherm B. (2006). GUMO – the General User Model Ontology. Available At: http://www.inf.unibz.it/~ricci/ATIS/papers/UM05_Gumo.pdf

[hohpe2006] Gregor Hohpe, Bobby Woolf (2006). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional. ISBN-13: 978-0321200686

[iso24751p1] ISO/IEC 24751-1: Information technology — Individualized adaptability and accessibility in e-learning, education and training — Part 1: Framework and reference model

[iso24751p2] ISO/IEC 24751-2: Information technology — Individualized adaptability and accessibility in e-learning, education and training — Part 2:“Access for all” personal needs and preferences for digital delivery

[iso24751p3] ISO/IEC 24751-3: Information technology — Individualized adaptability and accessibility in e-learning, education and training — Part 3: “Access for all” digital resource description

[keates2001] Keates S., Langdon P., Clarkson P. J., Robinson P., 8 April 2001. User Models and User Physical Capability. USER MODELING AND USER-ADAPTED INTERACTION, Volume 12, Numbers 2-3, 139-169. DOI: 10.1023/A:1015047002796

[mattson2005] Timothy Mattson, Beverly Sanders, and Berna Massingill (2005). Patterns for Parallel Programming (First ed.). Addison-Wesley Professional. ISBN-13: 978-0321228116

[mbui2010] Juan M. González Calleros, Gerrit Meixner, Fabio Paternò, Jaroslav Pullmann, Dave Raggett, Daniel Schwabe, Jean Vanderdonckt, José Manuel Cantera Fonseca (Editor). Model-Based UI XG Final Report. W3C Incubator Group Report 04 May 2010. http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/

[mda] Object Management Group (OMG) Model Driven Architecture (MDA). http://www.omg.org/mda/

[metsker2006] Steven John Metsker, William C. Wake (2006). Design Patterns in Java (Software Patterns Series). Addison-Wesley Professional. ISBN-13: 978-0321333025

[omgcat] OMG. Catalog of OMG Modeling and Metadata Specifications: UML. http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML

[omglinks] OMG: Useful Links to UML resources. http://www.uml.org/#Links-General

[paterno2005] Paternò F. (2005). Model-based Tools for Pervasive Usability. Interacting with Computers, Elsevier, May 2005, Vol. 17, Issue 3, pp. 291-315.

[paterno2009] Paternò F., Santoro C., Spano L.D. (2009). MARIA: A Universal Language for Service-Oriented Applications in Ubiquitous Environments. ACM Transactions on Computer-Human Interaction, Vol.16, N.4, November 2009, pp. 19:1-19:30.

[potel1996] Mike Potel (1996). MVP: Model-View-Presenter. The Taligent Programming Model for C++ and Java. http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

[raggett2008] Dave Raggett. SVG, layered user interfaces and end to end models. http://www.svgopen.org/2008/papers/59-SVG_layered_user_interfaces_and_end_to_end_models/

[rdf2004] Beckett D. (ed.) (2004). RDF/XML Syntax Specification (Revised), W3C Recommendation 10 February 2004. World Wide Web Consortium. Available at: http://www.w3.org/TR/rdf-syntax-grammar/

[reenskaug1978] Trygve Reenskaug. MVC XEROX PARC 1978-79. http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

[schmidt2000] Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschmann (2000). Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects. Wiley. ISBN-13: 978-0471606956

[shalloway2000] Alan Shalloway, James R. Trott (2000). Design Patterns Explained: A New Perspective on Object-Oriented Design (2nd Edition). Addison-Wesley Professional. ISBN-13: 978-0321247148

[siegel2005] Jon Siegel (2005). Introduction to OMG's Unified Modeling Language. http://www.omg.org/gettingstarted/what_is_uml.htm

[sosnov2007] Sosnovsky S., 13 March 2008, Ontological Technologies for User Modeling. Available At: http://www.sis.pitt.edu/~sergeys/papers/SOA_Sosnovsky_revised.pdf

[sun2002a] Sun Microsystems (2002). JSR-000075 PDA Profile for the J2METM Platform (PDAP). Available at: http://jcp.org/aboutJava/communityprocess/review/jsr075/

[sun2002b] Sun Microsystems (2002). Mobile Information Device Profile (MIDP); JSR 37, JSR 118. Available at: http://java.sun.com/products/midp/

[thiessen2007] Peter Thiessen and Charles Chen (2007). Ajax live regions: chat as a case example. In Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A) (W4A '07). ACM, New York, NY, USA, pp. 7-14. http://doi.acm.org/10.1145/1243441.1243450

[thiessen2010] Peter Thiessen and Stephen Hockema (2010). WAI-ARIA live regions: eBuddy IM as a case example. In Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A) (W4A '10). ACM, New York, NY. http://doi.acm.org/10.1145/1805986.1806030

[uaprof20] User Agent Profile version 2.0 (2006). OMA specification. Available at: http://www.openmobilealliance.org/Technical/release_program/uap_v2_0.aspx

[uml] Object Management Group (OMG) Unified Modeling Language (UML). http://www.uml.org/

[uml2011] UML Specification, version 2.4. http://www.omg.org/spec/UML/2.4/

[umldiag] Introduction to the Diagrams of UML 2.0. http://www.agilemodeling.com/essays/umlDiagrams.htm

[velasco2004] Velasco C A, Mohamad Y, Gilman A S, Viorres N, Vlachogiannis E, Arnellos A, Darzentas J S (2004). Universal Access to Information Services - the Need for User Information and its Relationship to Device Profiles. In: Gulliksen J, Harker S, Vanderheiden G (eds), Special issue on guidelines, methods and processes for software accessibility. Universal Access in the Information Society, 3 (1), pp. 88—95.[16]

[wai-aria2011] Lisa Pappas, Rich Schwerdtfeger, Lisa Seeman; James Craig, Michael Cooper (Editors). Accessible Rich Internet Applications (WAI-ARIA) 1.0. W3C Candidate Recommendation 18 January 2011. http://www.w3.org/TR/wai-aria/

[webIeeeIso] ISO/IEEE 11073-1020, Health informatics — Point-of-care medical device communication — Part 10201: Domain information model, 2004-12-15, Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1389297&tag=1

[webOWL01] OWL Web Ontology Language – Overview. W3C Recommendation 10 February 2004. Available at: http://www.w3.org/TR/owl-features/

[webRif] Kifer M., Boley H. RIF Overview. W3C Working Group Note 22 June 2010. Available at: http://www.w3.org/TR/2010/NOTE-rif-overview-20100622/

[webSkos] Miles A., Bechhofer S. SKOS Simple Knowledge Organization System. W3C Recommendation 18 August 2009. Available at: http://www.w3.org/TR/skos-reference/

[wiki_dpcs] Wikipedia: Design pattern (computer science). http://en.wikipedia.org/wiki/Design_pattern_%28computer_science%29

[wiki_mvc] Wikipedia: Model-view-controller. http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[wiki_mvp] Wikipedia: Model-view-presenter. http://en.wikipedia.org/wiki/Model-view-presenter

[wiki_uml] Wikipedia: Unified Modelling Language: Diagrams overview. http://en.wikipedia.org/wiki/Unified_Modeling_Language#Diagrams_overview

[16] DOI: http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s10209-003-0075-5

[zimmermann2004] Zimmermann G., Vanderheiden G., *et al.* (2004). Universal remote console standard - toward natural user interaction in ambient intelligence. 2004 Conference on Human Factors in Computing Systems, ACM Press.

[zimmermann2006] Zimmermann G., Vanderheiden G., *et al.* (2006). Universal Control Hub & Task-Based User Interfaces. URC Consortium. Available at: http://myurc.org/publications/2006-Univ-Ctrl-Hub.php

