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

Project no. FP7 – ICT – 258030

Deliverable D2.2.1 CARFO (R1)

SEVENTH FRAMEWORK PROGRAMME

Due date of deliverable: 31/08/2011

Actual submission to EC date: 31/08/2011

<table>
<tr><th colspan="3">Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)</th></tr>
<tr><td colspan="3">Dissemination level</td></tr>
<tr><td>PU</td><td>Public</td><td>Yes</td></tr>
</table>

Document Information	
Lead Contractor	Telefónica I+D
Editor	Telefónica I+D
Revision	v.1.0 (30/08/2011)
Reviewer 1	Fabio Paternò (ISTI-CNR)
Reviewer 2	
Approved by	
Project Officer	Jorge Gasós

Contributors	
Partner	**Contributors**
Telefónica I+D	**Javier Caminero Gil, Mari Carmen Rodríguez Gancedo**
CTIC	**Luis Polo Paredes, Diego Berrueta Muñoz, Sergio Fernández López**
SAP	**Safdar Ali**
UCL	**Vivian Motti**

Changes			
Version	**Date**	**Author**	**Comments**
0.1	20/07/2011	TID	First skeleton, introduction
0.2	11/08/2011	TID, CTIC, UCL, SAP	Intermediate version for internal review
0.3	22/08/2011	TID, CTIC, UCL, SAP	First Internal review comments
0.4	25/08/2011	TID, CTIC, UCL, SAP	First final draft
1.0	30/08/2011	TID, CTIC, UCL, SAP	Delivered version

Executive Summary

This deliverable presents the results of work undertaken in the FP7 SERENOA project in the topic of semantic knowledge representation for Service Front End adaptation. In it, we describe the overall fit of such a system in the general picture of the SERENOA architecture and runtime and discuss several topics related to the design and contents of the ontology itself. We present the results for the first module of the CARFO Ontology in the shape of an ontology module for Context of Use in the SFE adaptation domain. This includes the formalization of concepts relative to the User, the Platform and the Environment. We end the document with some conclusions that reflect upon the process of definition and some details upon future research in this task.

Table of Contents

1 Introduction 5
1.1 Objectives 5
1.2 Audience 5
1.3 Related documents 5
1.4 Organization of this document 6
2 CARFO Overview and role in the architecture 7
3 CARFO General Design 13
3.1 Relationship with the theoretical framework 13
3.2 CARFO and Runtime 14
3.2.1 Rule Execution Paradigms 14
3.2.2 Mixing Rules and Procedural Knowledge 15
3.2.3 From RDF to XML 15
3.2.4 RDF-driven XML Transformations 18
3.3 Modular Design of CARFO 19
3.4 Ontology Validation 20
4 CARFO R1: Context of Use Module 22
4.1 User 22
4.2 Platform 26
4.2.1 Capabilities 27
4.2.2 Device Structure 28
4.2.3 Device Models and Individual Devices 28
4.3 Environment 32
5 Conclusion & future work 34
5.1 Summary 34
5.2 Future work 34
6 References 36
Acknowledgements 38
Glossary 39

1 Introduction

1.1 Objectives

This document intends to set the foundation of one of the key elements of the SERENOA roadmap, which is the CARFO Ontology. This ontology is set to formalize the knowledge expressed in our theoretical grounding, such as our Design Space (CADS) and Reference Framework (CARF), initial versions of which have already been delivered for SERENOA. For this purpose, we will re-examine the state of this theoretical domain and provide guidelines for the ontology to be constructed.

In addition, CARFO will be the bridge connecting the theoretical foundations to more practical aspects of the SERENOA adaptation pipeline. To fulfill this, an additional objective of this document will be to explore how the fully working SERENOA system could take advantage of this ontology. For this matter, we will study the role of rules operating over the ontology and how they fit within the overall plan. An additional overview of how the runtime will be able to work with the knowledge in the ontology will also be presented.

The logical final aim of this work is to provide a formal ontology valid for SERENOA. At this stage the focus is on the Context of use, as one of the foremost domains to be taken into consideration. We will present the particulars of this ontology and how the concepts therein defined relate to the problems SERENOA wants to solve. This ontology will be independently checked for consistency and completeness.

It has to be noted that at this stage the discussion will necessarily tend to be in quite abstract terms, since we are in the process of elaborating tools (e.g., description languages for user interfaces and rules) that might prove valuable in providing more concrete examples than the ones given. This aspect will be addressed in future revisions of this document, when the aforementioned tools are more clearly defined.

In a nutshell, this approach will enable us to work in the more general aspects of the CARFO at this starting phase of the project. Our most prominent objective thus is to provide a grounded base upon which to incrementally build the rest of the ontology in following milestones of the project. Here our aim is to be thorough in defining what is the CARFO ontology, for what purpose it will be used and its main structure in broad terms.

1.2 Audience

The intended audience of this document is threefold:

a) First and foremost, the members of the consortium, who will find here a detailed account of what this part of the SERENOA environment is and what role it will play in the remainder of the project.
b) Secondly, as a publicly available document, the researchers in the relevant fields: adaptation of SFEs, semantic technologies and also medium-scale project software engineering.
c) Last but not least, the EC officials that will use the information in this document as an account of the activities taken in the project tasks that inform this work.

1.3 Related documents

In past project deliverable D2.1.1 'CADS and CARF (R1)' SERENOA introduced the foundations to the theoretical framework that will allow context aware adaptation of SFEs in the future prototypes. From that deliverable, work regarding this topic has been focused on two distinct areas.

The first is concerned with the transition of the theoretical framework into a computing one that will allow SERENOA to take advantage of it in real applications. Part of this work consists on representing the framework in a way that will allow the software entities to tap into the knowledge to perform meaningful adaptations. This, formalized as an ontology, will be described in this document. Other practical developments coming from this abstract framework will be discussed in D4.2.1 'Algorithm Library for AAL (R1)'.

The second area of improvement is related to the broadening and deepening of the theory itself. This will be the focus of the project deliverable D3.1.1 'Reference Models Specification (R1)' which will lay out a more refined theoretical framework. New versions of the implementation of this theory will be made available in deliverables that will expand the ones above mentioned, focusing on the refinements made.

1.4 Organization of this document

Chapter 1 presents the goal, audience and related documentation with this deliverable. In Chapter 2 the general outline of the CARFO Ontology is presented, along with a summarized description of the integration of it within the runtime data flow. Chapter 3 presents a detailed view of the CARFO design from several points of view, as its relationship with the theoretical framework (CADS/CARF), the usage of such an ontology for adaptation rules, the need for validation of the resulting work and its place within the SERENOA runtime. Section 4 presents the first part of the CARFO ontology developed for SERENOA, the Context of Use Module. Details are given for its three main aspects: user, platform and environment. Finally, Chapter 5 provides a set of conclusions for the work and proposes a list of future work lines for this topic in SERENOA.

2 CARFO Overview and role in the architecture

In this section we will provide a summarized account of what CARFO is and its expected role in the SERENOA project. We intend here to provide a more succinct view of the solution expressed in the project's Description of Work. There, references to the role of CARFO are many and distributed throughout the text to explain other particular aspects of SERENOA, but there is not a single unified description that provides an account of the necessity and, especially, of its role in the environment runtime.

Let us begin our discussion with a brief overview on the benefits of an ontology-based knowledge representation system in the field of adaptation. Studies in the literature such as (1) argue that an ontology-based approach is advantageous for the adaptation process, as compared to other state-of-the-art techniques, i.e. *filtering*. Adding further capabilities to the stack already available with XML, ontologies can be used to specify the semantics of data elements shared across systems. On one hand, the semantics defined with an XML schema are only available to the people that have specified it, while on the other hand, the semantics defined using Ontologies can be determined automatically by the systems at runtime (2). Therefore, the use of ontologies can address the lack of modeling data in current adaptive SFE systems. They can be used to share and reuse knowledge, and on the basis of the semantics formally specified in the accompanied ontologies, they can make sense of the data to exploit it for adaptation.

In the context of the SERENOA project, an ontology would be capable of capturing information about the user, the task, the system, the environment, and/or various aspects of the content (structure and presentation). This maximizes the amount of contextual information that can be used to accomplish sophisticated adaptation. Moreover, current adaptive service front-end systems rely on their own formalism and vocabulary for data representation. By the use of this as a standardized SERENOA-wide ontology, the systems can share and reuse model information to solve the inherent lack of data that hinders sophisticated adaptation. It will be used to define how to represent the semantic annotation of the information transferred and reused across the systems.

Complementary, the use of an ontology allows us to define and use *adaptation rules* that would improve the usability of SFEs by adapting them according to the current context (user, platform, environment, task, etc.) and making them more interactive. The advantage is that the adaptation rules can be intuitive and explicit; however their expressivity depends on the richness of the underlying data model. The more information provided by the data model, the more powerful and comprehensive the adaptation rules can be. These rules operate on the contextual information, which is stored in an *adaptation knowledge base*. This will be particularized to SERENOA and further explained in the reminder of this section.

There are many contextual dimensions that are relevant for the adaptation and CARFO ontology is used to represent them. The entire context being captured can then be regarded as a list of *conditions*. Given a relevant list, the system will generate the most appropriate *recommendations*. In other words, when a situation is recognized by the system, it will adapt the SFE accordingly. Modeling this behavior in the form of rules is indeed the core of our approach that we will use. The logic for adaptation is specified declaratively in the form of rules representing some generic *if-then* patterns. Such patterns i.e. a logic program is at a higher conceptual level than *if-then* statements in a purely imperative program. When adaptive behavior is captured by rules, the inference engines processing them produce recommendations that are more accessible to the user. The underlying inferences can be analyzed, provided with a proof and the rules can be made available for the user to be inspected and modified. This could allow for feedback loops, user-control and thereby enhance user's trust towards the system.

Having stated the rationale for the usage of ontologies in our work, we will now define CARFO as *an ontology for SERENOA that provides concepts for the representation of all knowledge present in the project's theoretical domain*. Namely, this refers primarily to the Design Space (CADS) and Reference Framework (CARF) (both already defined in (3)) and also others such as the Reference Models to be proposed at a later stage of the project in D3.1.1 and others. The name CARFO itself tends to be misleading as it seems to imply that it is an **O**ntology solely for the **CARF** Reference Framework. This is a simplistic view, as we do not intend to use the ontology only to define elements that may be part of the CARF, but to include concepts representing all elements in the field of adaptation. In Figure 1 we provide a bird's eye view of how it is placed in the SERENOA environment. The Figure mirrors the proposed technical and scientific approach to the project as proposed in the Description of Work:

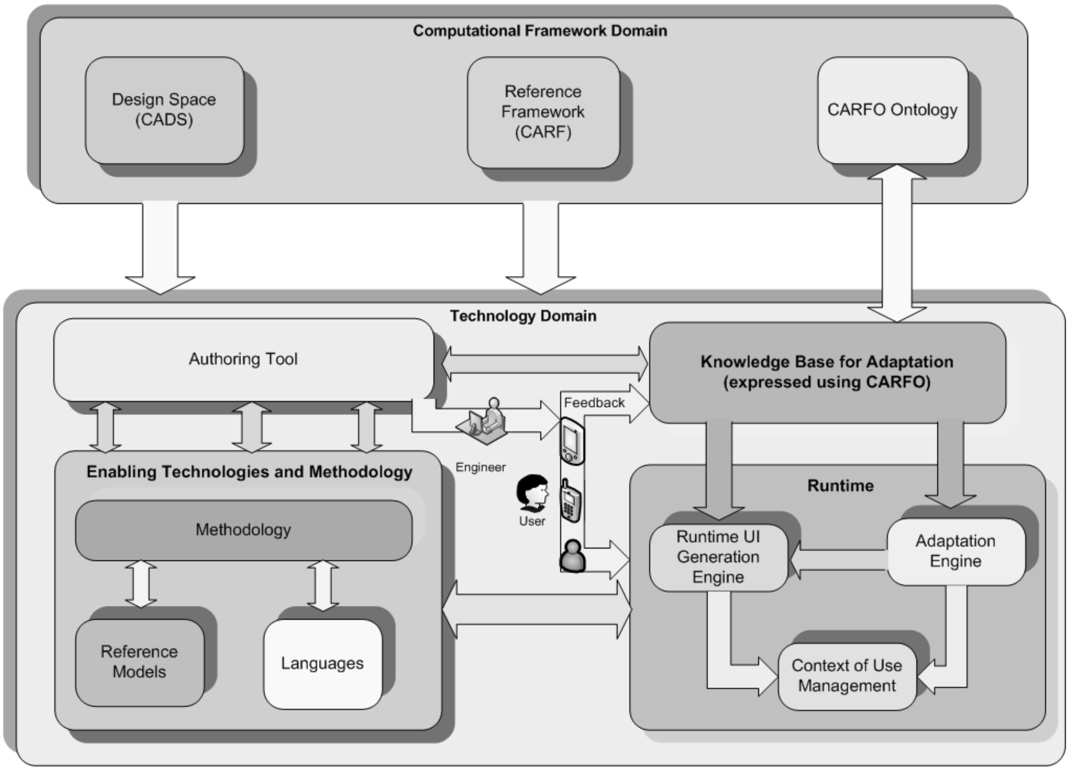
Computational Framework Domain
Design Space (CADS)
Reference Framework (CARF)
CARFO Ontology
Technology Domain
Authoring Tool
Knowledge Base for Adaptation (expressed using CARFO)
Feedback
Engineer
User
Enabling Technologies and Methodology
Methodology
Reference Models
Languages
Runtime
Runtime UI Generation Engine
Adaptation Engine
Context of Use Management

Figure 1: The SERENOA conceptual map

As it can be seen, there is a clear distinction between the theoretical and technical domains. However, we can see how CARFO, the subject of this document, involves both parts – in one case as the ontology itself and in the other as a fundamental part of the Knowledge Base that uses CARFO concepts. This makes clear the relevance of this work, as it will readily link the theory and practice in the project and become a grounding upon which the foundations of the adaptation technology to come will be constructed. Additionally, this dual presence makes it critical to clearly define the roles of the modules in the diagram and how they will relate to the software implementation. This section will explain these roles.

The three main building blocks of the computational framework are the following:

- **The Reference Framework** (CARF) identifying the relevant abstraction layers for the description of SFEs sensitive to the multiple dimensions of the *context of use*. To this aim, we intend to extend the Cameleon Reference Framework (4) (CRF) issued by the FP5 Cameleon project.
- **The Design Space** (CADS) identifying the relevant design options and how they can vary to accommodate different scenarios and requirements. The CADS will **support the whole adaptation life-cycle** by allowing each step being performed by the user, the system, a third party or any combination of them.
- **An Ontology** for Multi-Dimensional Adaptation of SFEs (CARFO) which will provide the semantic description, classification and relationships between the set of different concerns involved in **advanced adaptation logic** for SFEs. Such ontology will be the main formalism supporting the representation of adaptation knowledge.

Conversely, in the technology domain we have the Knowledge Base for Adaptation (henceforth, KBA), whose stated aim in the project description is the following

- The **knowledge base** represents advanced adaptation logic and expressed by means of the CARFO Ontology. It will contain (in a machine understandable format) all the necessary knowledge to perform multidimensional adaptation of SFEs. It will include all the rules, conditions and reactions to be performed in response to variations in the *context of use*

The Design Space (CADS) is a theoretical tool intended to support the whole adaptation lifecycle by allowing each adaptation step to be performed by the user, the system, a third party or any combination of them. In addition, the CADS can use feedback or adaptation preferences coming from the user or domain experts, thus enabling an adaptation system capable to learn from human beings. The CADS is fully described in (3) but summarily it can be said to be represented by a multi-dimensional space that is structured according, but not restricted to the following dimensions:

- The *means used for adaptation*: re-distribution, re-molding.
- The *UI component granularity* representing the smallest UI units that can be adapted by the way of these means.
- The *state recovery granularity* after adaptation has occurred (from the session level to the user's last action).
- The *UI deployment* (static or dynamic) as a way to characterize how much adaptation has been pre-defined at design-time vs. computed at runtime.
- The *context coverage* to denote the causes for adaptation with which the system is able to cope.
- The *coverage of the technological spaces* as a way to characterize the degree of technical heterogeneity supported by the system.
- The *existence of a meta-UI* to allow users to control and evaluate the adaptation process.

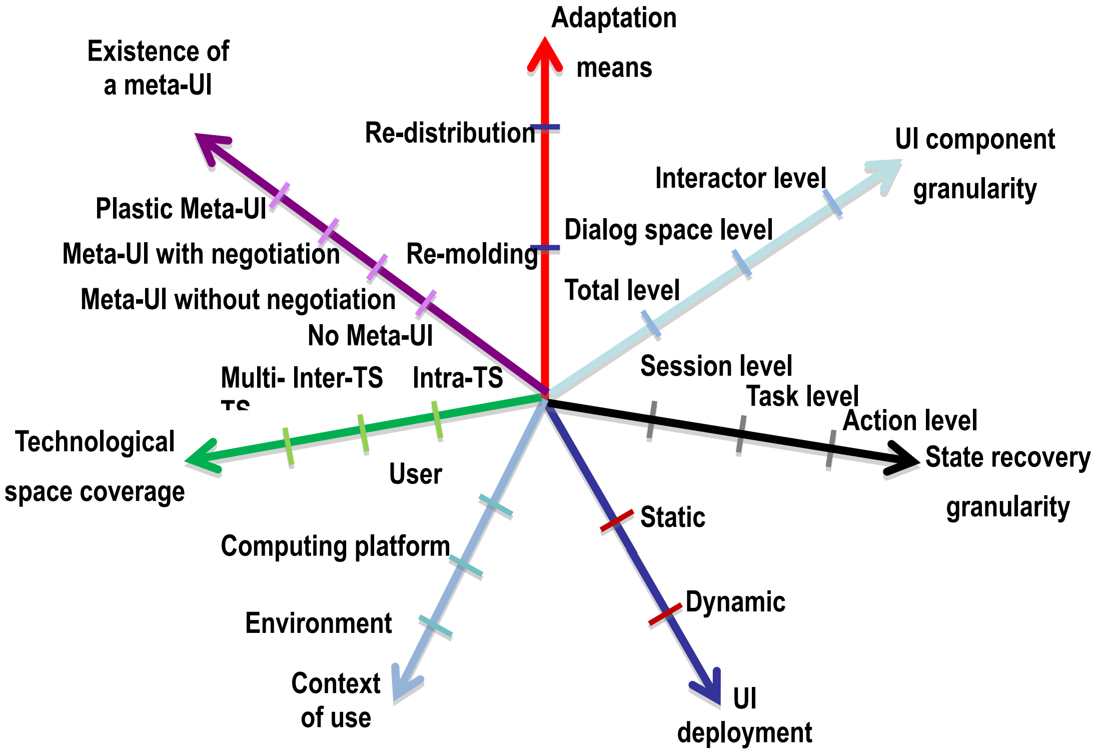

Figure 2: The Similar Adaptation Space [Vanderdonckt et al., 2007].

On the other hand, a Reference Framework (CARF) was elaborated and discussed also in (3) in order to articulate all dimensions, their abstraction levels and concepts relevant to adaptation. This was based on past work in the Cameleon Reference framework, which structures the UI development life cycle into four subsequent layers: *task and domain model*, describing the logical activities necessary to achieve users' goals; *abstract user interface (AUI) model*, *concrete user interface (CUI) model*, and *final user interface (FUI)*. The *abstract interface*, is the UI model independent of any target platform and any interaction modality; *concrete interface*, is the UI model which is dependent on the target platform and interaction modality, but independent of any implementation language, and the *final interface*, corresponds to the UI running or interpreted. The contents and usage of these abstraction levels remain however open to interpretation.

For the CARF, the content was gathered by performing the systematically analysing the literature: by compiling adaptation techniques, abstracting individual techniques into more general ones whether experience permits and consolidating techniques. The techniques were structured according to how, what, who, when, where, with respect to what and with which models. Finally the techniques were also generalized when permitted. Figure 3 below shows an example of a general element of such a reference framework.

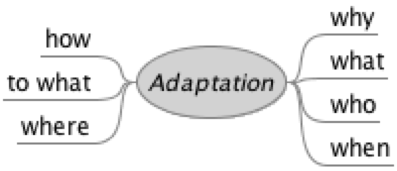

Figure 3. Reference Framework

More concrete instances of the framework are thus worked on by applying the general approach to a particular example of adaptation. As an example, the Figure 4 below is an example of a CARF for Text Content adaptation, expanding upon the template of Figure 3 for this particular type of content:

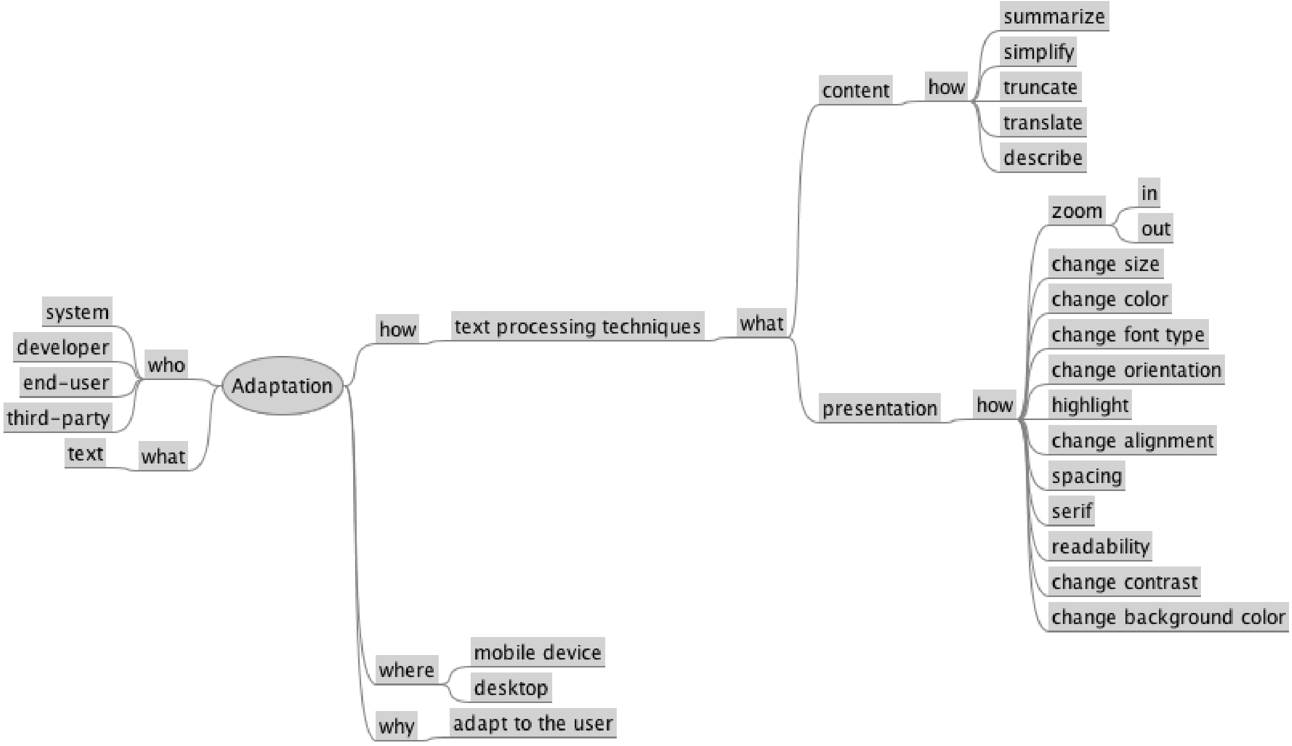

Figure 4. An example of CARF – Adaptations Techniques for Text Content.

In the chapter 3 of this document we will examine in more detail the available CARF descriptions at this stage of SERENOA. More specifically, as future work in CARF (D2.1.2) will address the association of the adaptation techniques with context information (mainly regarding: platform, environment and users), this information will be relevant to develop CARFO ontologies that will constitute the main output of this deliverable. CARFO should support SERENOA Adaptation Knowledge Base to compose adaptation rules that consider both issues (context and techniques),

To end this section, we will roughly examine how these ontologies will be usable by the rest of the runtime in order to articulate the context-aware adaptations that the SERENOA system is to implement. This is summarized in the following figure:

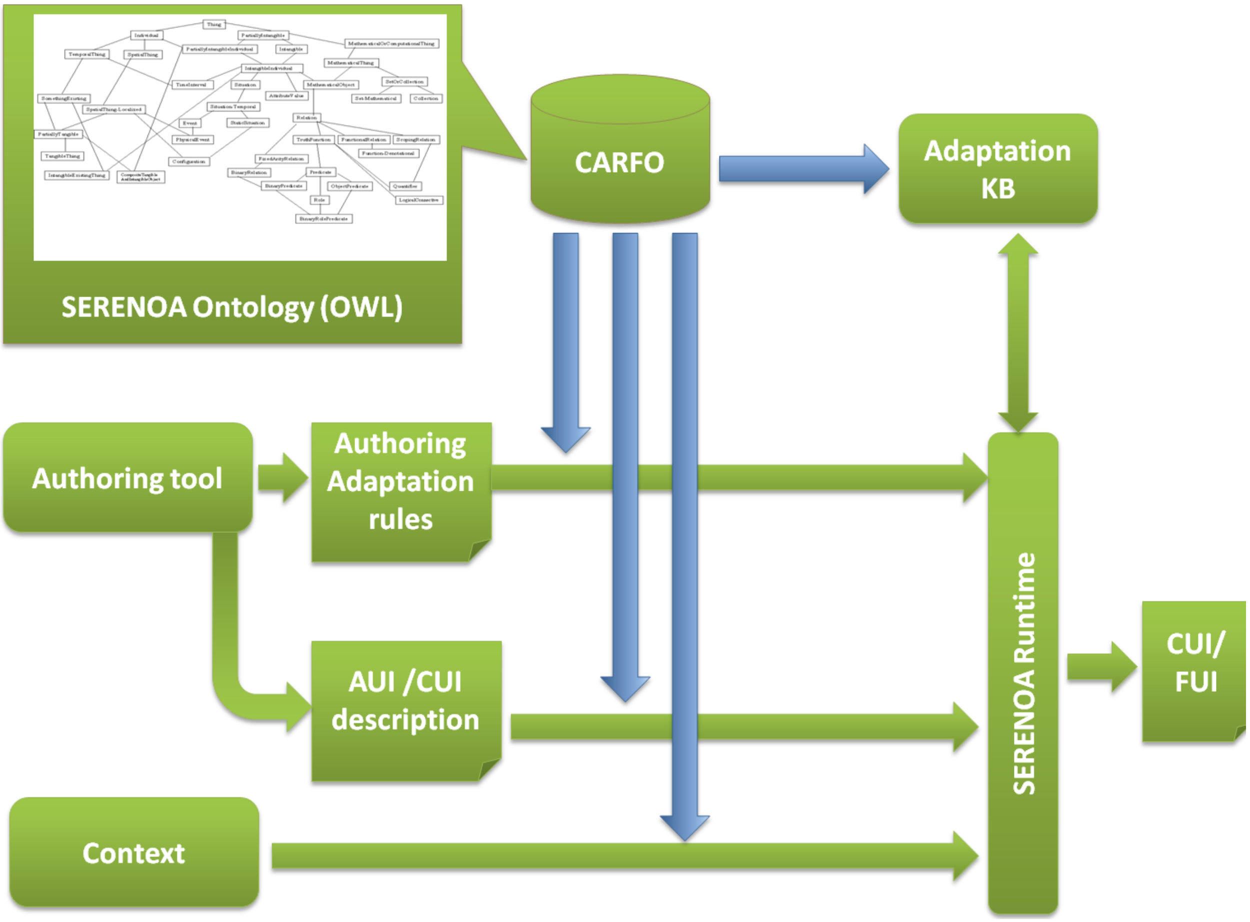

Figure 5: Adaptation data flow. Blue lines mean concept definitions and green lines are exchanged streams of data

In this Figure, a general data and concept flow for the adaptation process is sketched. A sample run through all the stages in the adaptation process will shed some light on the various roles that software entities will have to fulfil in the future of SERENOA. First, we can see how the Authoring Tool yields two separate descriptions:

- Abstract or Concrete UIs, which comprise the desired functionality of the UI (e.g., the user is to be presented with a series of independent interactive text inputs form and a submit button plus an erase function for the text box)
- preliminary set of adaptation rules that might have been defined by the developer (e.g., if the display is to be a mobile one, one of the text boxes will be omitted)

These AUI/CUI descriptions and Authoring Adaptation rules are coded in appropriate mark-up languages that are under development in SERENOA and which will provide descriptions relevant and flexible. These descriptions, as entities that could be adapted in future stages, will need to be described in terms compatible with the ontological description of the CARFO, as it is implied by the blue lines. In addition, the context will be captured at runtime by the Context Manager component in our architecture. This will describe both context information (platform, environment and users) and it also will have to be derived from concepts stated in the CARFO.

The next step is the most crucial as it is the adaptation itself. At some point the Adaptation KB retrieves data from the Ontology in the manner of an appropriate instance of a CARF model and augments the existing adaptation rules with it. In that way CARFO will support the Adaptation KB to define adaptation rules that could address the different space design dimensions considered in CADS. This can be thought as done in run time (e.g., the CARF for each case is calculated in the instants prior to the concretion of the AUI in a device) or in an offline mode (e.g., the ontologies have been running simulations of predefined use cases and a pre-recorded result is stored for each case). In any case the immediate result is that the Adaptation Rules are improved by this knowledge. This result is then passed on to the SERENOA runtime where the adaptations are performed according to the current context and Concrete and/or Final UIs are generated.

The above figure is nothing but a simplification of the process as some different aspects might be taken into consideration (e.g., the AUI or CUI not coming from a dedicated Authoring Tool but from a reverse engineering of an existing UI, the Adaptation KB gathering extra data from the AUI description). However, it is used to illustrate the overall role of the CARFO and KB for Adaptation and help the reader understand the more in-depth discussion over the next chapters.

3 CARFO General Design

3.1 Relationship with the theoretical framework

In general words the CADS (Design Space) presents possible alternatives that must be taken into account for the development of a context-aware application that supports adaptation. The CADS is a descriptive mechanism that permits applications to be analysed and compared according to the level of adaption that they support. As a Design Space it is a purely theoretical approach.

The CARF, Reference Framework, is an informative diagram in which the adaptive and adaptable applications can be defined according to the dimensions of adaptation that they consider. A Reference Framework provides also a set of indications for implementing adaptation, mainly regarding which techniques can be applied.

The CADS and the CARF are complementary approaches. They have many advantages, among which we can highlight the following: they are extensible (new dimensions can be accommodated), they are flexible (current dimensions can be more detailed, or removed) and they provide an integrated view of all concepts related to adaptation.

Although the CADS and the CARF aid developers and designers to support adaptation by presenting them many relevant concepts that must be taken into account while developing applications, they have some limitations:

- Relationships, such as constraints, cannot be established between the concepts that they presented
- They are not formalized (for instance defined with a standard modelling language, which could allow the support of their usage with tools and editors)
- No semantic information is presented

In order to formalize the concepts defined by the CADS and the CARF, Reference Models are being elaborated (they are reported and can be accessed in the D3.1.1). By using UML we formalize their definitions; this allows consequently the exchange and reuse of data.

Although UML is a widespread language to formalize models, it is restricted too. UML itself does not allow the description of constraints (OCL must be considered). Aiming to overcome these and also other limitations, the definition of an ontology is proposed, namely CARFO.

Regarding the concepts defined by the CADS and the CARF, an ontology may enhance them by:

- Providing semantic descriptions about the adaptation techniques (such as the information specified in the templates, see (3))
- Allowing inference and reasoning on top of the adaptation techniques defined (e.g. classifying the techniques according to the context information, or the abstraction level associated with them)
- Permitting relationships to be established (e.g. setting the compatibility level between techniques, associating similar techniques and strategies)

The information that an ontology defines may also be useful for implementing the machine learning algorithms proposed in the context of Serenoa. These algorithms, that are considered evolutive, are based on context information in order to select and apply adaptation techniques, methods and strategies that are more appropriate in a given context. The Ontology may aid this process by providing additional information not only about the context, but all the concepts involved, such as adaptation techniques, their relationships and constraints.

The inferences and constraints can also be associated with the software qualities preferred by users. In (5) the author performed a study to identify which aspects of the UI are preferred by users regarding consistency in multi-device UI. He points for instance that, changing the colour of the font affects more the consistency for the user than changing the font size or type. Four visual elements manipulation were analysed. Although he analysed these aspects only for multi-device UI, i.e., considering migration, which is one type of adaptation method, it is clear that usability guidelines such as this are also useful to prioritize adaptation techniques within other methods too.

3.2 CARFO and Runtime

CARFO will be delivered as an OWL2 ontology. OWL is the industry standard for web ontologies. The semantics of this language is based on a subset of First Order Logic, called Description Logic. This formalism is suitable to provide web-oriented rich, extensible data models, and offers a number of advantages over other modelling languages, such as UML. OWL2 ontologies permit to:

- Describe devices, interfaces, software platforms, networks and their components.
- Identify and profile the user and her/his preferences.
- To describe environmental and contextual conditions that may affect interface generation and adaptation processes.
- To reuse third-party, application-independent domain models. For instance, an ontology for e-business such as GoodRelations.

Description Logic essentially supports one operation, called reasoning. This operation enables model consistency checking, which is a task that is usually associated to design time. It also permits queries and sub-classification, which can be useful at runtime, e.g., to detect the class of a device given its characteristics. However, the expressivity of these queries is limited and does not reflect dynamic knowledge, such as the Adaptation Logic (AL) needed for adaptation and interface transformation. This kind of knowledge is typically captured by rules.

The combination of rules and description logics to efficiently building knowledge-based applications has been studied in recent years and some proposals have been made in order to augment rules with web ontologies written in OWL2. The FP7 ONTORULE[1] project is making significant advances in the state of the art of working combinations of both knowledge representation paradigms. This project suggests that rules and ontologies have different purposes and lifecycles; therefore they must be kept separate, even if at the end they are executed in combination.

In the remaining of this section, we discuss rule paradigms and their runtime execution. We illustrate rule execution with examples of interface adaptation and transformation rules. We envision three main uses of rules on top of the CARFO ontology:

1. RDF-to-RDF rules. These kind of rules implement logic within an RDF database. These rules can be used to augment the expressivity of DL ontologies. We envision that these rules can drive transformations between the higher levels of interface description in the CADS framework (for instance, from tasks and domain models to abstract interface descriptions).
2. RDF-to-XML rules. Eventually, most final and concrete interface languages are based on XML. These rules enable mappings between RDF and XML structures.
3. RDF-driven XML transformations. Wherever a transformation between two XML-based languages is needed, this kind of rules can be used to drive the transformations exploiting the knowledge in RDF databases. These rules are suitable for the lower levels of the CADS framework, allowing transformations from concrete to final interface languages.

3.2.1 Rule Execution Paradigms

Rules are expressions of the form "*IF a and b, THEN c*", where the IF part is called the conditions or body of the rule, and the THEN clause is called the head. The intuitive semantics of these expressions means that, given a set of facts F, *c* can always be derived whether *a* and *b* are present in F. There are two main rule execution paradigms. On the one hand, logical rules are based on subsets of First Order Logic (such as Horn rules). This rule paradigm has been widely studied in academic environments, and it is an active area of mathematical and artificial intelligence research. A representative example of this paradigm is PROLOG. These rules are used for constraint-satisfaction and goal-oriented problems, such as planning, theorem proving or games. Some extensions of logical rules have been proposed to meet industry requirements. One of the most well known ones is the Frame Logic, which is basis of the commercial product OntoBroker.

On the other hand, production rules implement operational semantics based on forward chaining of rules and

[1] http://ontorule-project.eu/

pattern-matching algorithms like RETE. A production system is comprised by a set of rules, a database, sometimes called working memory, which maintains data about current state, and a rule interpreter. These rules are used in the industry to implement expert systems, based on languages such as OPS5 or CLIPS. Currently, production rule systems are a technology widely adopted by companies to describe their business workflows and procedures, bridging the gap between business experts and IT departments and supporters. The main advantage of production rules versus logical ones is the scalability and performance. Even if the consistency of knowledge is sometimes lost in huge rule-based applications, production systems can deal with hundreds of rules and knowledge bases populated with thousands of facts. This primacy is also a hindrance, as the major difficulty is how to provide efficient means to manage this vast knowledge. Leading vendors of production rule systems are IBM (JRules), JBoss (Drools) and Oracle (Oracle Business Rules).

RIF is a W3C recommendation that defines a language to interchange rules in the web (6). The language comes in different flavours: a shared "core" dialect sits at the intersection of logic and production rules, and specializations for logic rules (RIF-BLD) and production rules (RIF-PRD). RIF is an XML-based language and it defines its semantic compatibility with other W3C web standards, namely XML Schema, RDF and OWL. All RIF dialects use URIs to identify resources. In the case of RID-BLD, the compatibility with OWL enables complex combinations of rules and ontologies reusing OWL axioms within the reasoning process. Moreover, OWL2 defines a profile (a subset of the language) that is intended to be implemented by any rule engine compatible with RIF-Core.

The following code illustrates a RIF-PRD rule that captures the logic of the statement: "The interface in the Galaxy Tab will capture the name of the person within a text area block".

```
Forall ?comp1 ?interface1 ?device ?interface2
such that And( ?interface1 # ex:AbstractInterface
            ?interface1 [ex:hasPart  -> ?component]
            ?component # ex:TextComponent
            ?component [ex:capures -> foaf:name]
            ?interface1 [ex:generate -> ?interface2]
            ?interface1 # ex:ConcreteInterface )
If  ?device # ex:GalaxyTab
Then Do ( (?comp2 New() )
       Assert ( ?comp2 # ex:TextArea )
       Assert ( ?interface2 [ex:hasPart -> ?comp2] ) )
```

Rules are executed using a rule engine or a *reasoner*. There is not any RIF rule engine, because RIF has been designed to be a lingua franca between rule engines and not an executable rule language. Therefore, rules must be written or exported to other (proprietary) languages. At the moment of this writing, support of RIF by commercial and open source rule engines is still limited.

3.2.2 Mixing Rules and Procedural Knowledge

Rules cannot represent all the knowledge needed for a given domain. Sometimes, part of the knowledge, such as image transcoding algorithms, must be captured by means of external procedures. Production rules make it easier to invoke these external functions, while logic rules usually suffer issues derived from the introduction in the domain of new (calculated) values that do not come from any reasoning process. Production rule engines usually allow calling procedural code written in programming languages (e.g., Java) as part of the rule firing. This feature is convenient for Serenoa because it permits to overcome the expressivity limitations of pure rule-based approaches to represent transformations. Moreover, it becomes possible to re-use existing algorithms and libraries.

For instance, a Java-based production rule engine such as Drools and JRules empower developers to integrate their Java code in their rules. Similarly, PyKe enables developers to do the same thing with Python code.

3.2.3 From RDF to XML

This section presents a proposal, available in the current state of the art, to bridge the gap between RDF and

XML languages. The former is the W3C recommendation of choice for modelling, interlinking and merging data in the web. RDF is used to provide resource descriptions reusing vocabularies provided by OWL and RDF(S) ontologies, being CARFO (possibly extended with domain ontologies) the target scope of this document. On the other hand, final (e.g., HTML) and concrete (e.g., IDEAL) interface description languages are XML-based ones. Transformations between both worlds enable moving from semantic, abstract design of interfaces in CARFO to concrete specifications to be rendered by concrete.

Since data in RDF is considered on a higher level of abstraction than XML data, the translation from XML to RDF is often called *lifting*, while the opposite direction is called *lowering*. Both translations are considered in Serenoa. On the one hand, the runtime must be able to generate specifications of final interfaces in an XML-based language. On the other hand, the reverse translation is also of importance, as strategies and methodologies to go from final to abstract interface descriptions will be examined in the context of the project.

The RDF language distinguishes three sets of disjoint syntactic entities, URIs, blank nodes (existential variables) and literals (data values such as floats or strings). An RDF graph G is a set of triples, where the tuple *(s p o)* is called an RDF triple. In the tuple, *s* is called the subject (it is an URI or a blank node), *p* the predicate (a URI) and *o* the object (one of the three syntactical units of RDF). A number of RDF graphs serialization syntaxes has been proposed: Turtle, Notation 3 (N3), RDFa, etc. However, the normative and recommended syntax is RDF/XML, which uses XML as the underlying representation model. This enables the use of XSLT and other XML tools to translate between RDF/XML and other XML formats. However, such transformations are considerably complicated by the flexibility the RDF/XML format offers for serializing RDF graphs. Thus, handling RDF/XML as XML data (and not as a graph comprised of set of triples) needs to consider different possible representations (i.e., a graph may be represented by alternative trees).

XSPARQL is an integration framework, proposed in (7), for combining XML and RDF, which relies on XQuery and SPARQL languages (8). XQuery and SPARQL are languages for XML and RDF query processing and transformation respectively. XSPARQL provides concise and intuitive mappings between XML and RDF in both directions and it is not affected by the graph serialization problem. This transformation language is a merge of SPARQL capabilities into XQuery. This way, XSPARQL benefits from SPARQL facilities to retrieve RDF data and from its Turtle-like syntax for constructing RDF graphs, while still having access to all the features from XQuery for XML processing.

Basically, queries in both languages are comprised in two parts: (a) the retrieval part (or the body); (b) the result construction part (or the head). XSPARQL enables the combination of these components in a unified language, where XQuery's and SPARQL's heads and bodies may be used interchangeably.

```
@prefix nfo: <http://www.semanticdesktop.org/ontologies/nfo/#> .
@prefix foaf: < http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://www.example.org/image_foo.png>  rdf:type  foaf:Image ;
                                        nfo:width  "140"^^xsd:integer ;
                                        nfo:height  "30"^^xsd:integer .
```

Consider the following RDF graph, where an image is described with certain width and height.

The following XSPARQL rule might be used to transform images descriptions into (X)HTML code. Notice that the application of this mapping will firstly generate a <div> element. Afterwards, an image environment is created referencing the image and declaring its size.

```
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
declare namespace nfo= "http://www.semanticdesktop.org/ontologies/nfo/#";
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace xsd = "http://www.w3.org/2001/XMLSchema#";
<div>{
for $depiction $width $height
from <example.rdf>
where { $depiction  rdf:type  foaf:Image ;
                                nfo:width  $width ;
                                nfo:height  $height .}
return
     <img  src="{$depiction}" width="{$width}"  height="{$height}"   />
}</div>
```

More complex *lowering* transformations are also allowed, such as using XSPARQL to generate RDFa documents. RDFa is a set of attribute-level extensions to XHTML for embedding rich metadata (RDF triples) within web documents. Thus, consider again the above RDF description of an image, augmented with a link (foaf:depiction) to the person it portraits.

```
<http://www.example.org/me>  rdf:type  foaf:Person ;
                   foaf:depiction  <http://www.example.org/image_foo.png> .
```

The following XSPARQL transformation captures this relation between the image and the person by means of the RDFa attribute rel="". Notice that to ensure that the triple representing the relation between the image and the resource <http://www.example.org/me> is correctly generated, the person has to be explicitly declared in the div environment as the subject of the subsequent triples (about=""). Working with a complete set of XSPARQL mappings, this is no longer necessary; as some of these mappings would be in charge of automatically declaring this information.

```
declare namespace (…) ;
<div about="http://www.example.org/me"> {
for $depiction $width $height
from <example.rdf>
where { $depiction  rdf:type  foaf:Image ;
                                nfo:width  $width ;
                                nfo:height  $height .
             $person  foaf:depiction $depiction .}
return
     <img  src="{$depiction}" width="{$width}"  height="{$height}" rel="foaf:depiction"   />
}</div>
```

A Java prototype implementation of the XSPARQL language that consists of translating each XSPARQL expression into XQuery expressions with interleaved calls to a SPARQL engine is publicly available[2]. The architecture consists of three main components: (a) a query rewriter, which turns an XSPARQL query into an XQuery; (b) a SPARQL engine (for the moment, it uses ARQ API of the Jena framework), for querying RDF from within the rewritten XQuery; and (c) an XQuery engine for computing the result document. This implementation builds upon the Saxon implementation of XQuery.

[2] http://sourceforge.net/projects/xsparql/

3.2.4 RDF-driven XML Transformations

It is often the case that concrete and final languages for interface descriptions are based on XML. Thus it is necessary to transform between XML documents. For instance the MyMobileWeb platform uses XSLT technology to this goal. More precisely, it is used to transform IDEAL interface descriptions into HTML code (or other final languages).

These transformation rules depend on the information of the context, which is often hardcoded within the rules themselves. In Serenoa, this context information is part of the CARFO ontology, and therefore it is available as RDF data. XSLT+SPARQL allows embedding SPARQL queries within the XSLT transformation templates (9). Therefore it is possible to benefit from CARFO to drive the transformation between XML languages.

The following example shows how context information can be used to drive the transformation from an IDEAL <image> tag into an HTML <img> tag. In this case, the XSLT rule detects the horizontal resolution of the device screen and automatically adjusts the dimensions of the image in order to fill the screen horizontally. Furthermore it generates an alternative label based on the title of the image. The context information as well as the description of the original image, is available as the following RDF:

```
<rdf:RDF>
  <rdf:Description rdf:about="http://example.org/myPhoto.png">
      <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Image"/>
      <nfo:width>800</nfo:width>
      <nfo:height>600</nfo:height>
      <dc:title>My photo</dc:title>
  </rdf:Description>

  <rdf:Description rdf:about="http://example.org/myCurrentDevice">
      <dco:horizontalResolution>400</dco:horizontalResolution>
      <dco:verticalResolution>400</dco:verticalResolution>
  </rdf:Description>
</rdf:RDF>
```

The XSLT file contains two templates. The first one matches IDEAL <image> tag and executes a SPARQL query retrieving the data from the RDF file. Then it forces the execution of the second template on the bindings obtained from the SPARQL query. This second template extracts the values from the binding to variables and then it simply creates the HTML tag and sets its attribute values, generating a result as follows: <img src="http://example.org/myPhoto.png" width="400" height="300" alt="My photo"/>.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                        xmlns:results="http://www.w3.org/2005/sparql-results#"
        xmlns:xalan="http://xml.apache.org/xalan"
        xmlns:sparql="XalanExt"
        extension-element-prefixes="sparql"
        version="1.0">
  <xalan:component prefix="sparql">
    <xalan:script lang="javaclass" src="xalan://net.berrueta.xsltsparql.XalanExt"/>
  </xalan:component>

<xsl:template match="image">
    <xsl:variable name="sparqlQuery" select="concat('
                        PREFIX nfo: &lt;http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#&gt;
                        PREFIX dco: &lt;http://example.org/deliverycontextont/&gt;
                        PREFIX dc:  &lt;http://purl.org/dc/terms/&gt;
                        SELECT ?hRes ?w ?h ?title
                        FROM &lt;http://dl.dropbox.com/u/881344/serenoa-data.rdf&gt;
                        WHERE {
                          &lt;http://example.org/myCurrentDevice&gt; dco:horizontalResolution ?hRes .
                          &lt;', @src, '&gt; nfo:width ?w ; nfo:height ?h ; dc:title ?title .
        }')"/>
    <xsl:apply-templates select="sparql:sparql($sparqlQuery)/results:results/results:result[1]"/>
  </xsl:template>

<xsl:template match="results:result">
    <xsl:variable name="title" select="results:binding[@name='title']/results:literal" />
    <xsl:variable name="w" select="results:binding[@name='w']/results:literal" />
    <xsl:variable name="h" select="results:binding[@name='h']/results:literal" />
    <xsl:variable name="hRes" select="results:binding[@name='hRes']/results:literal" />
    <img src="http://example.org/myPhoto.png">
      <xsl:attribute name="width"><xsl:value-of select="$hRes"/></xsl:attribute>
      <xsl:attribute name="height"><xsl:value-of select="$h * $hRes div $w"/></xsl:attribute>
      <xsl:attribute name="alt"><xsl:value-of select="$title"/></xsl:attribute>
    </img>
  </xsl:template>
</xsl:stylesheet>
```

3.3 Modular Design of CARFO

CARFO scope aims to cover a wide range of aspects, from description of interfaces to people and hardware. Due to this heterogeneity, Serenoa proposes a modular design of the ontology that permits to manage each module independently. Moreover, one of the goals is to reuse existing knowledge pieces in order to maximize interoperability.

OWL2 enables to assemble knowledge from different ontologies, scattered in different files (or web locations). References can be made by URIs, or even by the owl:imports mechanism. In addition, OWL2 is represented as RDF graphs, which have the nice property to be seamlessly merged regardless of their origin. Figure 6illustrates both mechanisms to combine separated ontology modules in OWL2. A common pattern in this approach is to have a high-level ontology orchestrating several smaller ontologies.

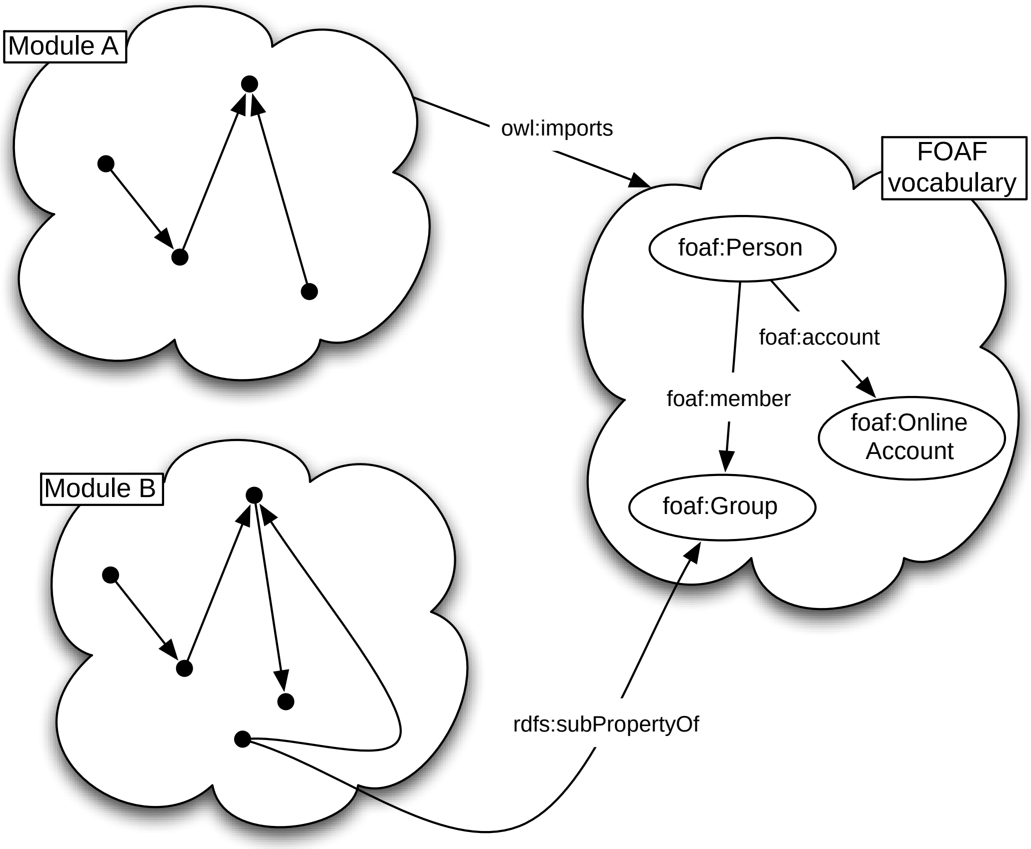

Figure 6. Example of a modular ontology

A 1-to-1 mapping from CADS axis to CARFO modules is not envisioned. This is due to the different granularity of CADS axis. For instance, the Context of Use axis is large enough to be considered not only as just one module, but as three (as it will be discussed in Section 4). On the other hand, the UI deployment axis, which indicates whether the deployment is static or dynamic, does not have the same complexity. Nevertheless, the list of CARFO modules is not closed yet and subsequent versions of this deliverable will address this issue.

3.4 Ontology Validation

In SERENOA, we will produce several versions of the full CARFO Ontology during the course of the project. It is our intention to periodically check its relevance with regard to our system (e.g., the architecture, adaptation techniques and use cases) and also its internal consistency to ensure that it won’t cause problems when using it for knowledge representation and/or reasoning. This work will be described in detail in future deliverables (e.g., D2.3.1 ‘CARFO Validation R1’ and D2.2.2 ‘CARFO R2’) that will have complete versions of the ontology available. In this sub section, we will briefly summarize the different validation strategies and tools that we are studying for future use.

Formal ontologies are fundamental for the Semantic Web and play a pivotal role in cases where a shared ontology suffices multiple purposes, i.e. crafting rules, populating knowledge base etc. However, in case where the systems semi- automatically perform the reasoning over the knowledge base to draw conclusions, it is essential that the shared ontology be consistent. If an ontology is inconsistent, then false conclusions may be deduced. According to the OWL model-theoretic semantics (10), an ontology is consistent if there is an interpretation that satisfies all the facts and axioms in the ontology. Such an interpretation is called a *model* of the ontology.

In order to check and validate the Ontologies, the OWL Test Cases document (11) defines an OWL consistency checker as follows:

“An OWL consistency checker takes a document as input, and returns one word being *Consistent*, *Inconsistent*, or *Unknown*”.

There are a number of systems used for ontology consistency checking (12). The tableaux reasoned searches for a model through a process of *completion*. The tableaux completion starts by constructing an initial completion graph from the ABox. The nodes in the completion graph intuitively stand for *individuals* and *literals*. Each node is associated with its corresponding *types*. Property-value assertions are represented as *directed edges* between nodes. The reasoner repeatedly applies the tableaux expansion rules until a clash (i.e. a contradiction) is detected in the label of a node, or until a clash-free graph is found to which no more rules are applicable.

The OilEd (13) ontology editor is intended for small-scale ontology development and consistency checking, though it is not a complete ontology development environment. OilEd uses FaCT (14) for its consistency checking, which is a description logic classifier that can also be used for modal logic satisfiability testing. It can check the consistency of a DAML+OIL ontology, but it cannot check DAML+OIL itself. Furthermore, one can use FaCT only if one imposes some additional limitations on a DAML+OIL ontology that go beyond those of DAML+OIL itself. For example, one cannot have a cardinality restriction on a transitive property. Finally, FaCT only checks consistency, it does not issue warnings that indicate possible mistakes that are not inconsistencies in themselves. JTP (15) is a theorem prover, which is developed in Java, accepts KIF axioms but it doesn't support paramodulation and only accepts axioms in *Horn clause* form. Since the DAML+OIL axioms contain *equalities*, *equivalences* and other non-Horn structures, it is not compatible with DAML+OIL.

Chimaera is a software system that supports users in creating and maintaining distributed ontologies on the web (16). Two major functions it supports are merging multiple ontologies together and diagnosing individual or multiple ontologies. It supports users in such tasks as loading knowledge bases in different formats, reorganizing taxonomies, resolving name conflicts, browsing ontologies, editing terms, etc. While the Chimaera system is an effective tool for ontology integration, its diagnostic suite is currently limited and not connected to a full theorem prover (17).

Pellet (18) is not only a complete OWL-DL consistency checker and a very incomplete OWL-Full consistency checker, but also an OWL syntax checker. Pellet is the first and currently the only complete OWL-DL consistency checker and has the most coverage of OWL as a whole of any reasoner (though some reasoners, particular OWL-Full ones, cover areas of OWL-Full reasoning Pellet just does not try to handle).

4 CARFO R1: Context of Use Module

The context of use is an important part of the CARFO ontology, and comprises three elements: the user, the platform and the environment. Each element plays a role in the concrete and final levels of the interface generation. The information about the user may impact on the design of the interface in order to accommodate her special needs (e.g., accessibility requirements); the information about the platform describes the constraints that must be taken into account to tailor the interface to the parameters of particular devices; and the environment provides contextual information such as location, brightness, languages, etc.

This part of the CARFO ontology is arranged in modules, which can be individually used. Some of these modules, such as the one that describes users, are built upon existing ontologies and vocabularies. This section discusses related work and how it relates to the CARFO design principles and requirements.

The most relevant effort to design an (OWL) ontology with the same purpose as CARFO was the W3C Delivery Context Ontology (DCO) (19). This ontology aimed to model devices, networks and environmental conditions that may impact on content adaptation, particularly for the mobile web. The DCO presents some expressivity and modelling limitations, in particular regarding its extensibility, modularity and its ability to describe artifacts at different levels of abstraction; moreover, it contains some expressions that are not idiomatic in OWL. In addition, there is not any known implementation. W3C discontinued support for this effort as part of an internal re-arrangement of their activities. CARFO plans to build on the experience of DCO, as well as to use it as foundation for the device and platform description.

For the sake of simplicity, in what follows it is assumed that the default namespace is the one of the Context of Use module. Concepts and properties from external ontologies use their most common prefix.

4.1 User

Users are a central pillar of the CARFO ontology, as interface adaptation must not only consider physical and circumstantial aspects (the platform and environment) but users' preferences and profiles of target applications. Some ontologies are already available, and have some popularity, to describe several aspects of users. CARFO ontology does not aim to replace these ontologies but to reuse them.

FOAF Ontology

The Friend-Of-A-Friend ontology was one of the first popular vocabularies in the web of data (20). It is a lightweight RDF(S) vocabulary. It provides a set of concept and properties to describe people and social connections. The main concept is obviously foaf:Person. Being an RDF resource, a foaf:Person is identified by means of a URI and it is characterized by attributes, such as name (foaf:name), mail (foaf:mbox), birthday (foaf:birthday), etc. The identity of a person in the web can also be described by her homepage or her accounts in the social websites (for instance, LinkedIn and Twitter).

On the other hand, FOAF also enables to capture relationships between people. It provides a generic framework based on an abstract property called foaf:knows. It is assumed that any social relationship between two persons is a specialization of this property, for instance, the connections between relatives, workmates and friends.

Furthermore, in combination with web certificates, FOAF can be used to build a portable web identity, called WebID. This is a decentralized architecture of identity management empowering users to identify themselves by URIs and to control the information they want to share with third-party websites and other users.

SIOC Ontology

The goal of the SIOC initiative (Semantically-Interlinked Online Communities) is to enable the integration of online community information (21). SIOC is based on an OWL ontology for representing rich data from the social web (blogs, social networks, mailing lists, etc.) in RDF. In the past four years, SIOC has recently achieved significant adoption through its usage in a variety of commercial and open-source software applications. SIOC is designed to be fully compatible with the FOAF vocabulary for expressing personal profile and social networking information.

Current online-community sites are isolated from one another. The potential synergies among many sites, communities, and services are expensive to exploit, and their data are difficult and cumbersome to link and reuse. For a instance, parts of the answer a person is looking for are implicit in various discussion of a number of online communities, but people participating in one discussion can't readily access information about related discussions elsewhere. The main reason for this lack of interoperation is that for the most part in the social web (from Facebook and Twitter to private chats), common standards still don't exist for knowledge and information exchange and interoperation. SIOC's ultimate goal is to fill this niche providing an RDF-based format for social data exchanging.

The central concept of the ontology is sioc:UserAccount, which captures a given user in an online community site. A foaf:Person is normally registered on a site through a user account. The property foaf:holdsAccount enables cross-links between people and their multiple accounts. Notice that the property sioc:account of allows establishing reverse relations between a sioc:UserAccount and the foaf:Person. SIOC also introduces a set of properties and concepts to provide descriptions about the user generated content, such as posts in a weblog and messages in a chat board, represented as instances of sioc:Post. Posts can also be threaded whether they are connected by a common subject or by reply. Furthermore, SIOC data model also covers the channels where these discussions are made (sioc:Forum), which can be linked to the web sites that host them (sioc:Site).

SIOC enhances the description of both forums and posts by means of SKOS concepts in order to create mediated links between user-generated content. These relations may enable the navigation between several kinds of resources, meeting the "linked data" initiative.

WAI Ontology

The Who Am I![3] (WAI) vocabulary aims to extend FOAF through introducing the concepts of roles and profiles. In the real world, people are more than just persons, they might be musicians, presidents of government, firemen, football players or car drivers in a traffic jam. Moreover, people modulate their personality to the pertinent situation or context. For instance, John as a member of the last.fm community expresses some musical interests, which can be used to find like-minded people and to recommend some contents (artists, genres, albums, etc.). John's preferences watching TV may be completely different, and it is necessary to capture this complexity inherent to individuals and their involvement in society.

WAI is an OWL2 ontology, designed to be fully compatible with FOAF and SIOC vocabularies. It provides a flexible mechanism for FOAF documents extension, intended to model people, specifying temporal and social features like their jobs, position in a company, tastes, security credentials or status in a given community. This mechanism is built upon two central concepts. On the one hand, WAI introduces the wai:Role class. A role is defined through a property that can be predicated of a person. In this ontology, roles are reified as first order individuals and relations between roles and players are expressed by means of the wai:plays property. WAI does not impose any *a priori* subclassification of roles. The concept is open to be refined according to domain or application requirements. However, as roles are considered instances, WAI comes up with the property wai:specializes, which enables the construction of role hierarchies, such as "student of philosophy is a sub-role of university student".

On the other hand, WAI also introduces the concept of wai:Profile, where profiles are entities capturing the dynamic and temporal aspects of roles. The full meaning of a sentence such as "John was the sales department manager of big company from 2000 to 2007" cannot be represented by a simple relation between John and the role "sales department manager". Profiles are introduced to cover this knowledge representation gap. Roles are not inherent to people, as they are not essential properties. A wai:Profile is a mechanism that allows referring to people when they are actually playing a given role, i.e. "person-as-role". As it occurs with sioc:UserAccount, profiles might seem that introduce a multiplication of identities for the user as well as an increase of resources to describe a particular individual. Nevertheless, all the profiles gather together at the foaf:Person. The multiplication of identities is only apparent. Moreover, profiles are resources identified by URIs. From this perspective, profiles and roles ontological distinctions contribute to data description enrichment according to the linked data paradigm.

Profiles are also useful to represent users in different contexts, introducing them as a mechanism to provide

[3] http://purl.org/wai

conceptual coordinates for "contextualizing" both roles and profiles. However, no more assumptions are made about its interpretation. A natural extension of wai:Context in CARFO is made by means of temporal and geographical locations. This is a typical scenario for personalized applications and interfaces, for instance in the field of ambient intelligence or mobility, where the adequacy is made considering several relevant aspects of the user.

In addition, social communities and on-line services can be also considered as contexts for profiles. On the one hand, communities are connected groups of people which are usually materialized in the Web as generic social networks like Facebook or LinkedIn, but also as dedicated on-line communities arisen from specific web sites: last.fm for music contents and FilmAffinity for movies are good examples for this case. In conjunction with FOAF and SIOC, social communities representation could benefit from WAI profiles management. When contexts and communities are used to fix the interpretation coordinates of the profile, roles may be implicit. In this case, a profile is considered a "person-at-context" or a "person-in-community", rather than "person-as-role".

RECO Ontology

Preferences are an important part of user profiles for many applications and user-oriented tasks. Preferences are statements of the form “Alice prefers A than B” or “Alice thinks A is better than B”. Basically, preferences are user modal attitudes about objects and situations of the world. Despite a considerable number of proposed languages for representing user preferences, effective publication of this information in an exchangeable format is far from being a reality.

One of the most successful proposals is the Review vocabulary, a lightweight OWL ontology intended to capture rating and reviews. However, this vocabulary lacks the expressiveness of complex preferences, such as “the user prefers terminals with high screen resolution” and “Alice likes Metallica albums, but not Jon Bon Jovi ones”. Other efforts have been made to represent user preferences in particular domains. The CC/PP vocabulary is a W3C initiative expressing device capabilities and user preferences to guide the adaptation of delivered content (22). However, CC/PP preferences are limited to desired attributes of device components. In addition, it is necessary to eliminate expressivity restrictions in CC/PP, which do not exist in RDF. CC/PP defines a hierarchical structure based on two main levels (components and attributes). It means a significant restriction over RDF, as its expressiveness is considerably reduced. In practical terms, CC/PP could be seen as a kind of big table in the form key-value, where content providers are very restricted in what regards to the semantic relationships they can define. In the multimedia domain, an OWL ontology has been proposed to specify how to combine content filtering and browsing criteria with Boolean operators. These kind of preferences are designed to be applied to MPEG7 and MPEG21 specifications (23).

A closely related initiative is (24), where a language for preferences for querying databases is defined. This approach introduces an algebra and operators to capture “wishes” of users. This formal, abstract language is then translated to extensions of SQL and XPath for relaxed queries. As it is not guaranteed that there will exist exact matches for all the conditions of a given query, preferences allow looking for the best possible matchmaking.

The RECommendations Ontology[4] (RECO) proposes language for representing and exchanging preferences as RDF data in the web. Moreover RECO is domain-independent, and it is designed to be compatible with any other existing RDF(S) and OWL vocabulary, as well as domain objects described as linked data (for instance, DBpedia resources). RECO distinguishes between two complementary notions of preferences:

1. *Preferences-as-constraints*, i.e., conditions about preferred attributes of the resources. A constraint is a set of qualitative descriptions of the desired attributes that objects must ideally satisfy in order to be of interest for a user.
2. *Preferences-as-ratings*, i.e., quantitative measurement of the “appealingness” of a particular object to a user. A rating captures the user satisfaction with respect to a given object within a scale of numerical values (also called utility). Recommendation systems typically use the real interval [-1,1] for calculating final utilities, while application front-ends measure users’ opinions with discrete scales, like the five stars classification used by Amazon or YouTube.

[4] http://purl.org/reco

RECO introduces the concept reco:Preference which reifies the relation between a user profile and a constraint. This relation is realized by means of the property reco:holds. Moreover, the auxiliary concept reco:Pattern is provided in order to capture the constraints comprising a given preference. For instance, a preference such as "Alice like crime films starred by people born in Spain", shown in Figure 7, is built upon two complementary constraints. The first constraint is crime films. The second one refers to actors born in Spain. The concept reco:Pattern is the mechanism that enables reusing other domain ontology, in this case the DBpedia vocabulary, providing integrity to the complete preference expression. This mechanism allows compositionally building complex constraints in RDF.

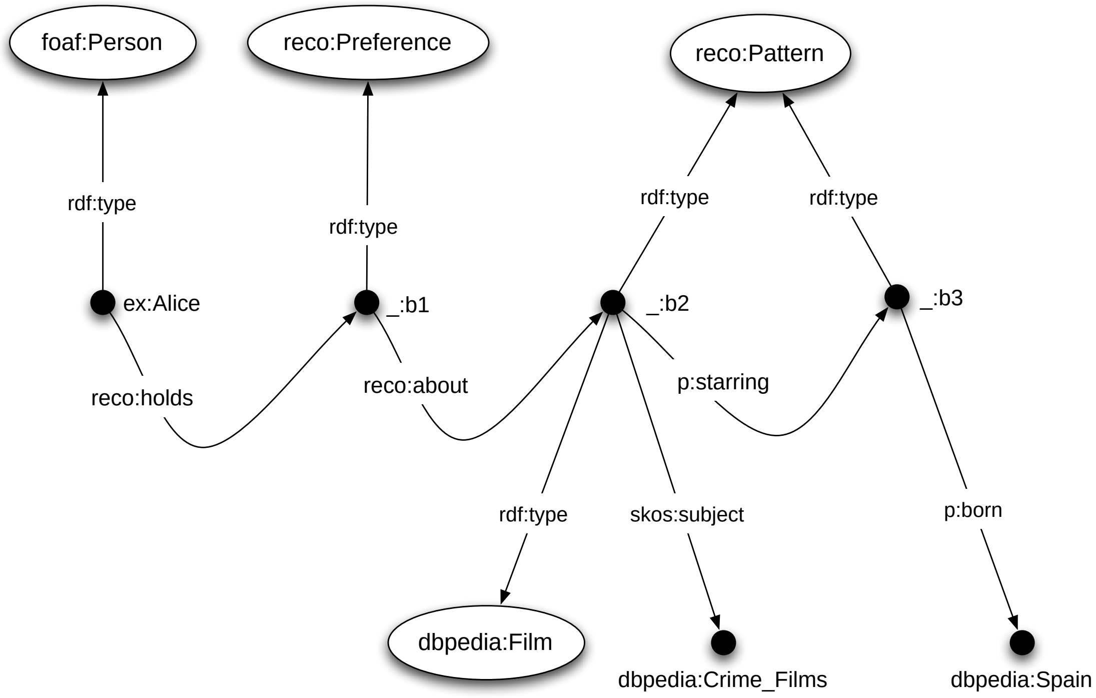

Figure 7. RECO representation of "Alice like crime films starred by Spanish actors"

RECO vocabulary also provides machinery to express constraints on data values, for instance "prize under 180€". Operator nomenclature is reused from XPath specification in order to ensure interoperability.

Regarding ratings, a ternary relation between a user, a item (any RDF resource, including instances of reco:Preference) and an utility value, the ontology introduces another concept, reco:Rating, which reifies this tuple. Three specific properties capture the relationships between the rating and the user (reco:assignedBy), the item (reco:rates) and the utility value (reco:utility).

RECO plays an important role in CARFO design as it enables the representation and exchange of user preferences across applications and scenarios. CARFO may also benefit from RECO preferences as they can be translated to rule and query languages, thus these preferences can be of interest of, on the one hand, recommendation systems that may filter content to be presented in final user interfaces. On the other hand, Serenoa runtime might consider these preferences for the interface generation and adaptation process, using these preferences to accommodate applications front-ends to specific need and profiles of users. Furthermore, being RECO a domain-independent vocabulary opens the door not only for a reutilization of preferences themselves, but for reusing existing domain ontologies that actually serve as data models for some given applications and services.

This section has presented four fully compatible OWL vocabularies to be reused in the context of the CARFO ontology for user representation. Reusing existing data models embraces the linked data and semantic web initiatives and facilitates data exchange and interoperability among applications and servers. This way, Serenoa builds on top of widely adopted vocabularies that can be considered as *de facto* standards, and widens its scope outside the limits of the project and the consortium members.

As an overview, Figure 8 illustrates the compatibility of the above-presented vocabularies. Notice that an application can combine and extend concepts and properties, adapting them to its specific requirements.

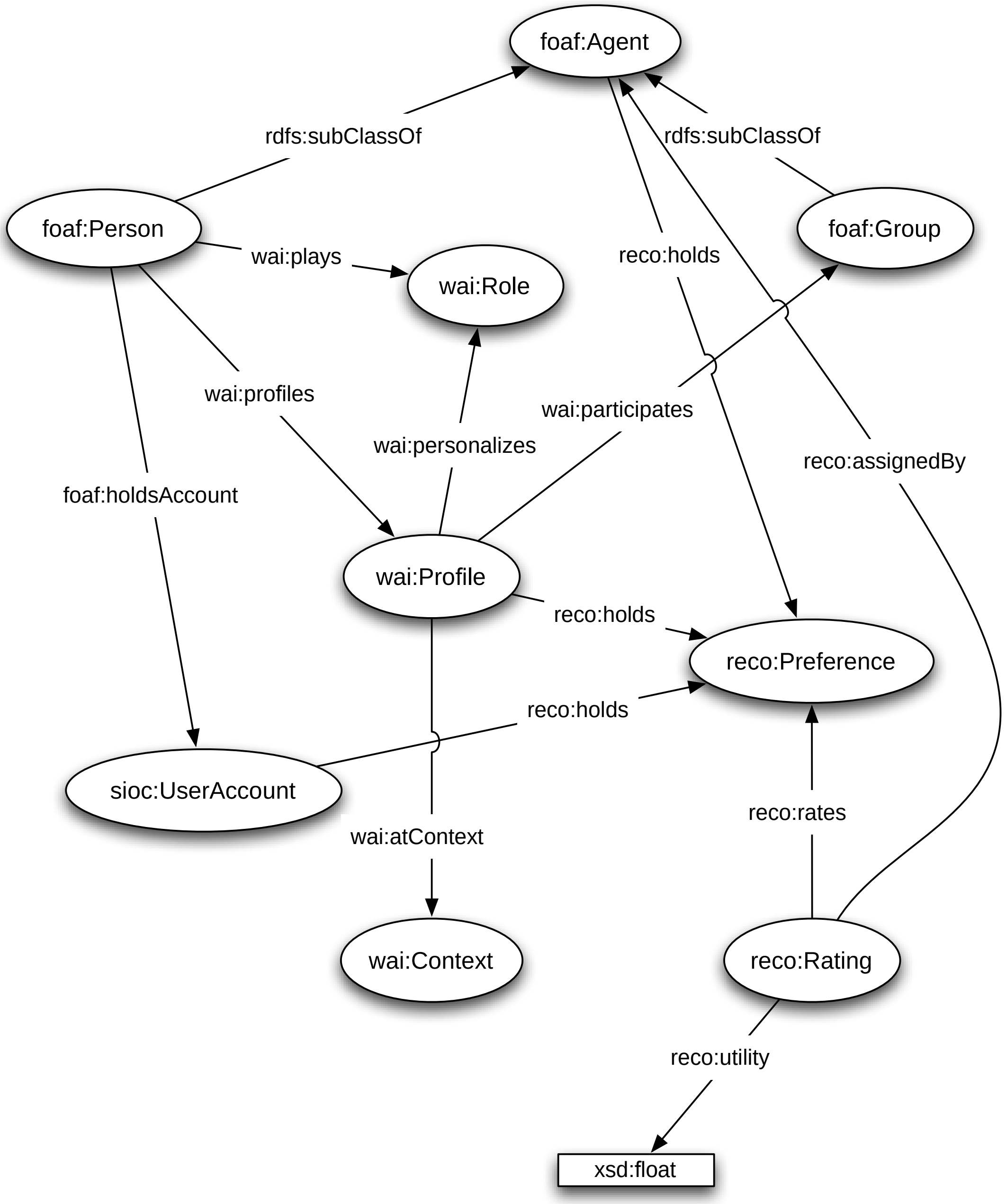

Figure 8. Combination of user-oriented ontologies in CARFO

4.2 Platform

In the CARFO ontology, the term “platform” comprises all the elements of hardware, software and networks relevant to describe a computing environment. This is a much broader concept than just a server or client side framework, such as MyMobileWeb or Android. Therefore a comprehensive description of the platform is challenging, because it involves different abstraction levels, a wide range of devices and complex software applications, such as web browsers and operating systems.

This deliverable puts the focus mostly on device description as a starting point, exploring the design space and state of the art solutions.

UAProf is a vocabulary proposed by the Open Mobile Alliance from the CC/PP specification defined by the

W3C through the extinct Device Independence working group, which is expressed in RDF. UAProf profiles are created as documents expressed in the homonymous vocabulary. They are referenced by means of a URI provided by some web browsers (generally, a significant amount of mobile web browsers) in their HTTP requests. Both UAProf and CC/PP offer an interesting framework for device description (other well-known databases include WURFL, DeviceAtlas and Alembic). They have been providing device descriptions used by the software industry over the last decade. Hundreds of device models expose their software and hardware characteristics by means of a UAProf document which may be cached and then enhanced.

It is important to note that UAProf documents contain static device descriptions. This means that they contain information that is known *a priori*, such as screen resolution, Bluetooth profiles supported or the web browser(s) installed from factory. Some examples of dynamic device information are battery charge level, or screen orientation (landscape, portrait).

Previous research work has considered UAProf and CC/PP limitations. The first efforts in the analysis of these specifications (25) (26) reflect the absence of a formal specification for profile resolution, the lack of a mechanism to allow combining profiles expressed in different vocabularies and the need for a formal definition of vocabularies –unfortunately, often indicated as comments in UAProf profiles.

In (27), a CC/PP-based vocabulary is proposed in order to represent more detailed context information for content and software adaptation. One of the most relevant problems found in CC/PP is the organization of device description in two layers. This forces the use of undesired syntactic sugar to express some definitions and relationships between device properties.

Related to the aforementioned work, an interesting study of the limitations in CC/PP and UAProf is presented in (28). Its conclusions state that basing CC/PP on RDF does not seem very appropriate as it basically models a hash table with name-value pairs. The study also considers that CC/PP and UAProf describe the data structures in which device profiles are represented but they do not provide an API to access the properties contained.

Sometimes, information about a generic software component (for instance, web browser) is provided in a UAProf description. Actually, a device may include more than one implementation of that software component (for instance, two or more web browsers) without its UAProf clarifying whether both of them are compliant to the description. UAProf lacks the ability to express these implementation details.

CARFO device descriptions will build on the experience of UAProf, and will seek backwards compatibility in order to make use of the extensive existing descriptions available in UAProf repositories. The remaining of this section addresses three design issues related device description. We describe generic solution patterns that do not just apply to device description, but to all the platform components.

4.2.1 Capabilities

Hardware and software components from the same vendor or the same family usually share common features. It is just natural to organize the information to minimize redundancy by creating descriptions that refine or extend other descriptions. The representation of these relations and their semantics are not straightforward in RDF.

In some cases, the different values that an attribute in UAProf may take are strictly defined. This leads to incoherent device descriptions in the sense that, for instance, an attribute is valued with a string for which there is no formally specified format. As an example, consider the values for the attribute *BluetoothProfile* in a set of actual UAProf description files downloaded from different manufacturers' repositories. A quick read after the values accepted show that the support for the AVRCP Bluetooth profile is noted with different strings (in alphabetical order): *"audio video remote control"*, *"Audio Video Remote Control Profile"*, *"Audio Video Remote Control"*, *"Audio Video Remote Control – Target"*, *"Audio Video Remote Control Profile"*, *"audiovideoremotecontrol"*, *"AVRCP"*, etc.

Some other typical inconsistencies include the expression of the values of a same attribute by means of different types. Following the same procedure for the *BluetoothProfile* attribute, a study of the *NumberOfSoftKeys* attribute has been carried out. In addition to the expected *xsd:integer* values (0, 1, 2, 3, 4, etc.), a "None" value has been found for many Motorola devices, such as the A1600. This is due to different versions of the CC/PP schema, UAProf vocabulary and third-party schemas (such as those from the 3GPP) published over time.

Ontology languages on top of RDF, such as RDF Schema and OWL, bring in the ability to declare certain constraints. However, some of the aforementioned integrity constraints are beyond the expressiveness of these languages. Even those that are possible usually lead to consequences that are not intuitive for people trained in databases and XML. Thus, it may make sense to introduce an *ad hoc* validation tool that implements the logic behind semantic restrictions. Finally, UAProf documents offer many chances for improvements in the light of recent developments in linked data (29). More specifically, UAProf documents, profiles and the resources they contain should be assigned HTTP-resolvable URIs. By doing so, they become extensible and linkable, and new opportunities appear for re-using shared resources.

4.2.2 Device Structure

A device is an aggregation of components, based on a blueprint. For instance, a mobile phone comprises a display, a keyword, antenna, battery, CPU, memory, etc. Moreover, these components can usually be disaggregated into smaller parts, i.e., they are themselves artefacts.

The CARFO ontology introduces two properties to capture the structure of any artefact, including devices: hasPart (and its inverse isPartOf) and component (and its inverse componentOf). Both properties have similar semantics, enabling to capture the mereological[5] relationship between a given artefact and its constituents. The former is a transitive property, which is useful for some purposes but as a consequence the order relation is lost in the case of complex hierarchy. On the other hand, the latter is suitable to keep separated the different levels of the hierarchy. Notice that both properties are anti-symmetric. In other words, two distinct entities cannot each be a part of the other.

This solution is inspired by the upper-level ontology DOLCE, which identifies a set of properties to represent different mereological relationships (such as temporal and spatial inclusion). Other ontologies, like Dublin Core, also provide machinery to capture the relation between a whole and its parts, in this case, applied to information resources (multimedia documents).

4.2.3 Device Models and Individual Devices

One of the modelling challenges of the CARFO ontology (and artefacts in general) is the distinction between concrete products and product-types (also known as models). The usage of a single class, for instance to capture devices, leads to some confusion and inconsistencies as detailed below:

1. **Indiscernible resources**. It is not possible to semantically distinguish between one instance representing a device model and another representing a particular device because both individuals inherit the same properties from the class they belong to (i.e. Device).
2. **Incoherent updating policy**. Particular devices are subject to changes: (a) due to the dynamical evolution of the context, such as the battery level, the signal power and environmental parameters (location, temperature, etc.); and (b) users might personalize properties of their mobile phones. A mobile phone might be connected with other devices within its local environment, such as an external display, a print or just another mobile device. On the other hand, models are invariant descriptions of types of products, where updating rules only apply when an evolution of the product is place on the market.
3. **Inconsistent descriptions**. There is not a mechanism to distinguish between models and particulars. Therefore, automatically checking the semantic consistency of the instances of Device is far from being a trivial task. For instance, consider a device model (such as the HTC Touch Diamond). If there is a contextual property (e.g. location) applied to this product-type, it is not possible to detect *a priori* that there is an inconsistency.
4. **Redundant information**. Apart from modelling issues regarding products and product-types, inheritance mechanisms are not clearly defined. CC/PP approach proposes a vocabulary where

[5] "Mereology" is the "theory of parthood relations: of the relations of part to whole and the relations of part to part within a whole" as defined in the Stanford Encyclopedia of Philosophy

properties of device models are treated as default features, and the user (or developer) can overwrite these values. However, this implies that default properties of a device (i.e. factory defaults) have to be repeated for every terminal of the same model.

4.2.3.1 Meta-modeling and OWL

CARFO proposes a new concept Model to capture these collections of products. In this context a 'model' represents a type of a product, where product is understood as an artefact, i.e., anything that has been industrially created by a company and offered to the market. This way CARFO is able to distinguish between concrete devices, such the mobile phone, the laptop and the camera of John Phillips, and models of these products, namely an iPhone 3GS, a Sony Vaio and an Easyshare M590 Kodak Camera.

There are three distinct alternatives to formally capture the concept Model in an OWL ontology. The next paragraphs we analyse these approaches, discussing pros and contras of these representations. Figure 9 graphically presents these alternatives in the particular case of device modelling, which will be used as the running context for the discussion.

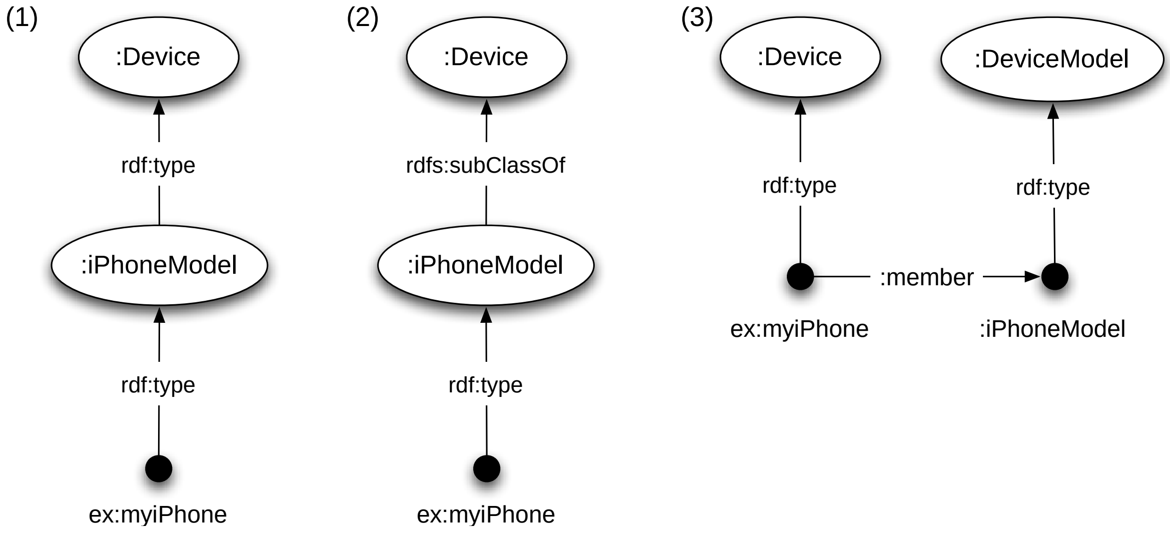

Figure 9. Proposals for modeling product-types

(1) "Metamodeling" Alternative:

A common practice in conceptual modelling is to separate the conceptual model from the data it describes. However, another approach is also possible, based on the separation of complementary layers in the conceptual model. Some of these layers play the role, depending on the point of view, of concepts and data, i.e. in a given knowledge base, there are data, types and types of types. This mechanism is known as *metamodeling*.

Regarding products and product-types, this means that both are considered first-order entities. In other words, they are domain resources with some attributes and properties. However, a product-type is also understood as a class, which permits to describe a kind of resources, namely, concrete products. Consider the concept Device (for instance, mobile phones) and the distinction between device models and concrete devices. The class iPhoneModel captures the product-type of the iPhone mobile device model. A particular iPhone, such as myiPhone, is an instance of this class, thus it inherits all the properties from it. Furthermore, iPhoneModel is also considered as an instance of a high-level class, Device, which actually captures generic properties of device product-types.

OWL2 provides some mechanisms to represent metamodeling.

- OWL2 metamodeling support. The update of the OWL specification includes some new features and syntactic freedom. A given URI can be used in an ontology to refer to more than one type of entity: classes, properties and instances. A strict separation of elements of an OWL2 vocabulary is not required. Therefore, it is allowed to state facts about classes and properties themselves. Entities that share the same URI should be understood as different "views" of the same underlying notion.

Moreover, expressive Description Logics has also been proposed as the underlying semantics of the OWL2 language in order to provide inference background for metamodeling. Some OWL2-compliant reasoners already support for this new feature.

Nevertheless, metamodeling is not a widely adopted modeling technique, even if there are some available implementations. The problem is that the syntactic freedom is a computationally costly feature, and only works properly when descriptions of classes and properties are rather simple (not with complex constraints). On the other hand, it forces the usage of a DL reasoner to deal with it.

- A simpler approach is provided by the RDF-based semantics of the OWL language (also known as OWL2 *Full*). This case the semantics of the language is not logic-inspired, but an extension of the RDF semantics (D-entailment). This semantics provides a precise formal meaning for every RDF graph. In the RDF-based semantics of OWL2, individuals may play different roles. For example, an individual can be both a data property and an annotation property, since the different parts of the universe of an OWL2 RDF-based interpretation are not required to be mutually disjoint, or an individual can be both a class and a property by associating both a class extension and a property extension with it. The expressiveness of this language is very powerful, but it is not recommended for building an ontology, as the distinction between the data and the conceptual model is lost. There are only RDF graphs.

(2) "Product-types as classes" alternative:

The second approach this document analyses is representing product-types (models) as classes in OWL2. The difference with the above one is that product-types are subclasses of the top concept Model (Device, in the example), not instances. Models are then described using OWL2 class restrictions, such as "all iPhone 3GS have a capacitive touch screen and a resolution of 320 × 480 (HVGA) at 163 ppi". These properties are inherited by the all the specific mobile phones instances of the class iPhone3GS.

However there are two main problems with this modelling design. On the one hand, product-types cannot be described as entities of the domain, they are part of the conceptual model. In other words, product-types are used to describe the domain data, but actually they are outside of it. A model cannot be linked to other resources, except in annotations axioms. On the other hand, OWL2 description logics semantics is a subset of first order logic. This means that if an individual is declared to be instance of a given class, it (always) inherits all the properties from the class definition. This is a problem when a product deviates from the standard definition of its product-type, due to some customization of the production. For instance, the battery of a smartphone can be changed. In these cases, two descriptions (the inherited and the personalized) will be available at the same time, inducing inconsistencies in the final definition of the product.

(3) "Product-types as instances" alternative

The last approach we are considering to correctly capture product-types in CARFO is to reify product-types, which can extensionally considered as collection of products, as entities of the domain, i.e., as first-order individuals. This way both product-types and products are at the same level in the ontology. The former are instances of the class named Model (DeviceModel, in Figure 9), while the latter are instances of the class Product (Device, in the same figure). To retain the membership relation between a product and its product-type, the property memberOf is created.

The main disadvantage of this approach is that the instantiation relation between a product-type and its products is lost. The property memberOf does not have a special status in the ontology. Therefore, no inheritance mechanism can be used between the two entities. However, there are some benefits of this design pattern. First, product-types are part of the domain, without introducing metamodeling. It is possible describe a model as any other resource. Second, the design pattern provides RDF-queriables descriptions of product-types, which can be performed by means of the SPARQL standard query language.

Last but not least, this approach enables the reconciliation of factory defaults and customization features of a given product. Figure 10shows how a device personalization is combined with the description of its product-type using this design pattern. The descriptions of the iPhone4 and Nexus1 models cover their complete technical specifications: number of megapixels of the camera, operating system, battery, connectivity, etc. This means that a particular device should match this specification in order to be considered a member of this model, i.e, to be catalogued as an iPhone4 or a Nexus1. Figure 10 represents the display width of two devices (ex:device1 and ex:device2) that belong to different product-types. This information is needed to correctly adapt the delivery content to the phone, but the width value for the iPhone device is unknown. The goal is to retrieve this information from the RDF graph asking the value, in one case, directly to the terminal properties (the Nexus1), and in the other case, indirectly through its product-type specification (the iPhone4).

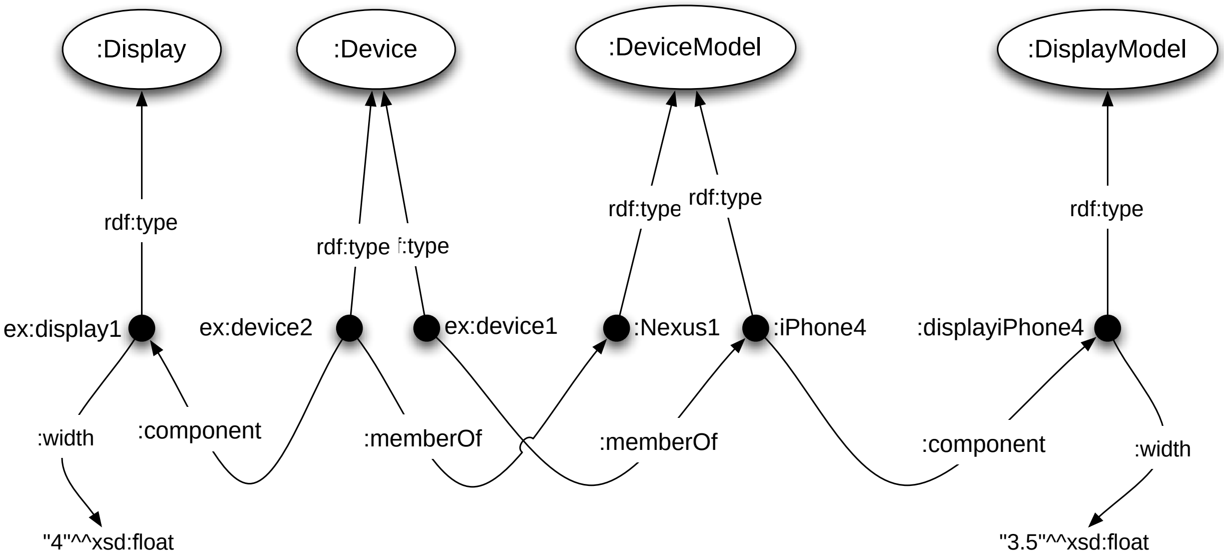

Figure 10. Example of a device description using the "product-type as instance" approach

We use SPARQL language to query the RDF dataset. The following SPARQL query demonstrates the union of two graph patterns. The query firstly tries to find whether there exists a known value of the terminal that matches the first graph pattern (i.e. the width of the display). If there isn't any known value for this property, the query will check whether the technical specification of the device's model does match the second graph pattern of the WHERE clause. This second pattern of the union expression simulates Negation as Failure behaviour, obtained by a complex graph pattern that combines an OPTIONAL and !bound FILTER expressions.

```
PREFIX ex: <http://www.example.org#>
PREFIX serenoa: <http://www.w3.org/2001/di/Group/Ontologies/DeliveryContext.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?device ?width ?model
WHERE {
{ ?device serenoa:component ?display .
  ?display rdf:type serenoa:Display .
  ?display serenoa:width ?width . }
UNION
{ ?device serenoa:memberOf ?model .
  ?model rdf:type serenoa:DeviceModel .
  ?model serenoa:component ?displaymodel .
  ?displaymodel rdf:type serenoa:DisplayModel .
  ?displaymodel serenoa:width ?width .
OPTIONAL {  ?device serenoa:component ?display .
                    ?display rdf:type serenoa:Display .
                    ?display serenoa:width ?specificwidth . }
FILTER (!bound (?specificwidth))
}
```

The execution of the SELECT query returns the results presented in Table 1. Notice that the ?model variable is bound only for those rows in which the default value is returned. This way, it is possible to distinguish between the default and concrete values returned:

?device	?width	?model
ex:device1	"3.5"	:iPhone4
ex:device2	4	

Table 1. Results of the product-type factory defaults query

Another relevant aspect of this design pattern of product-types is how sub-classification is accomplished. Consider for example the car model: "Volkswagen Golf", which is a car manufactured by Volkswagen since 1974 and marketed worldwide across six generations. This time long there have been many configurations of this car. Each of these configurations is a new product-type that is related with the general definition of the "Volkswagen Golf" model but introduces some modifications and specializations in the description of the model (engine, transmission, wheelbase, number of doors, etc.).

The hierarchy between product-types is easily captured when they are treated as classes, but not when they are individuals. Inheritance is not defined for domain data, i.e., the subclass relationship is not applicable to first-order entities. However, the specialization of product-types is relevant for CARFO because there are groups of device product-types that internally sub-specialize themselves. For example, for the iPhone 3G device model description, all device instances share the display size, the audio support, power and battery, etc. However, there are two available versions of the phone, one with 8GB flash memory, and another with 16GB. To this end, a new property specializes is included in the ontology. This property enables relations between product-types, i.e., establishing a partial order or hierarchy: if a product-type A specializes a product-type B, then product-type A shares the properties of B, but B doesn't share properties of product-type A.

4.3 Environment

The third pillar of the Context of Use module is the environment surrounding the user and the platform. In CARFO, the environment is interpreted in a broad sense, comprising not only ambient conditions (such as temperature, geographical location, time, etc.) but also contextual ones (such as "I'm at home" or "I'm busy at work").

Previous efforts to describe environment include CONON, for Context Ontology (30), which defines general concepts such as location, activity, person or computational entity. These terms are thought to be extensible in a hierarchical way by adding domain specific concepts. The authors divide their context model into an upper ontology and a specific ontology. On the one hand, the upper ontology is a high-level ontology that captures general features of basic contextual entities. On the other hand, the specific ontology defines the details of the general concepts and their features in each sub-domain covered.

Other significant works are related to ambient intelligence. As an example, the Amigo ontology was developed by the FP6 Ambient Intelligence for the Networked Home Environment project. This initiative enables the description of sensor networks and environmental profiles (amigo:EnvironmentalProfile), providing a preliminary set of measurable ambient conditions.

The CARFO ontology will embrace these previous works and extend them as necessary. A special solution will be developed for the precise representation of measurable quantities (not just for the environment, but also for the description of the platform). The motivation for this work arises from the limitations detected in the state of the art, namely the lack of adequate mechanism to capture semantics of measurements as RDF literals. We plan to extend MUO[6] (the Measurement Units Ontology) to measure environmental conditions, such as ambient temperature, or device properties, such as screen size. Notice that different measurement units are often used to measure these physical qualities (e.g., Fahrenheit and Celsius degrees, and inches, centimetres and pixels).

Figure 11 illustrates how to accommodate the representation of the surface area of Spain’s territory, according to MUO ontology.

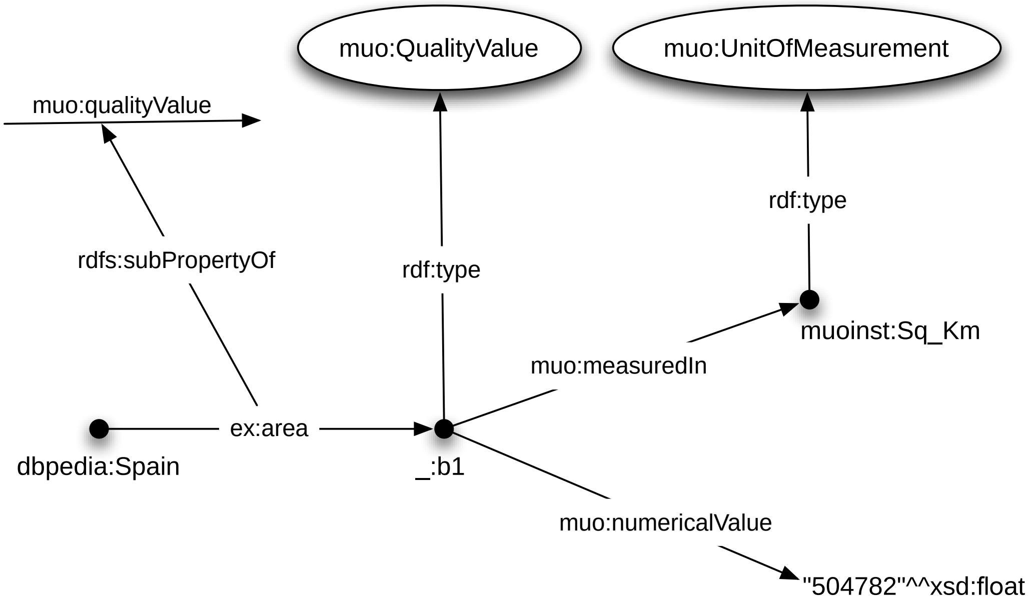

Figure 11. MUO representation of Spain's area

[6] http://idi.fundacionctic.org/muo/

5 Conclusion & future work

5.1 Summary

In this document we have presented a general overview of the CARFO Ontology. This included a thorough description of what will be its use in SERENOA in both the theoretical and runtime stages (section 1-2), an overview of several crucial ontology related issues (section 3) and a first approach to the actual details in the design of a part of the ontology (section 4).

Due to the complexity of the challenge it intends to solve (SFE adaptation), SERENOA needs a knowledge representation paradigm that will allow for future growth and cross-domain exchange of information. As discussed in past deliverable D1.2.1 and others, the nature of the adaptation is such that data (e.g., context of use) will be required to flow to different parts of the system in which it will be used for different tasks and/or software entities. This approach requires the usage of sound data formats, and we have found ontologies to be a solid basis upon which to build them. The CARFO Ontology will be used to define all the concepts necessary to describe our overall adaptation process and hence concepts defined in it will be used as common grounding to express among others adaptation rules, SFE description languages at different levels of abstraction and context information that may be relevant to trigger adaptation actions.

Due to this, the definition of the ontology is not an easy task. For this work, the consortium has taken the approach of first identifying the intended use of the ontology, then providing a reference albeit scaled down version of a working ontology and finally start the task of completing the ontology with all the needed concepts and modules. This document has covered the first (definition) and parts of the second (a reference downscaled implementation) objectives by providing a thorough description of the role of the ontology in sections 1-3 and then a first glimpse of actual ontology definition in section 4. Aspects such as validation and real-world integration of the ontology and related entities (e.g., reasoned) in the SERENOA framework have been just outlined and will be explored in future documents. This is summarized in the following subsection.

As a conclusion, we would like to remark that it is now clearer to our eyes what are the needs for such an ontology in our system and, most importantly, what are its limits for our purposes. Upon reviewing the DoW of SERENOA we detected that some affirmations therein contained (e.g., that the ontology would indeed contain the adaptation rules) were at least naive to some degree. Due to our design work in the semantics of the adaptation process and advancements in our study of the theoretical framework we now understand better what information needs to be modelled by the CARFO. This is undoubtedly the most significant result of the work described in this document and will help to steer further developments in many areas of SERENOA in the most useful directions.

5.2 Future work

The work undertaken in this topic and described in this document will be continued during the coming months of the project. The result of this work is to be documented in the following project deliverables:

i) D2.2.2 CARFO (R2)
ii) D2.3.1 CARFO Population (R1)

We will now examine in more detail the proposed lines of research for this continuation of work. In this document we have fulfilled two main objectives:

1) Present a thorough overview of how CARFO, as an ontology able to represent concepts in our theoretical work, will be able to assist in SERENOA-powered SFE adaptation.
2) Start writing the first module of the ontology itself, beginning with one of its most prominent aspects in adaptation, the Context of Use.

In 1), we have further set the stage for the very important stage of SERENOA that is beginning just after the writing of this document. In D1.2.1 (31) we introduced the technological elements that will conform our system in terms of modules and their relationship as a unified architecture. In future deliverables (D3.2.1, D3.3.1, D5.2.1) we will finish the preparation work and come forward with all the elements needed to develop a first working version of the SERENOA technology.

However, there are some preliminary future steps to be taken in the way of developing a software version of the CARFO and rules that might operate upon its concepts. Harking back to Figure 1 in Chapter 2 of this document, it can be observed that the way the CARFO is integrated in the runtime system is through a module there called the Knowledge Base. This will be the piece of software that a) contains the CARFO and any instantiated elements based upon it and b) is able to take the rules and provide reasoning upon these elements. The engineering of this Knowledge Base will be the next step to be taken in order to include the wealth of knowledge defined in the CARFO in the actual adaptation pipeline. Work on this is set to begin upon the delivery of this report and will be documented in the upcoming deliverables mentioned in the heading of this section, mainly in D2.2.2.

For objective 2), and regarding the ontology itself, we need to proceed in two distinct directions. The first is geared towards the completeness of the ontology with regard to the theoretical model in (3). More modules of the ontology, similar to the Context of Use, need to be specified, coded and validated using the same procedure here described. In addition, the CARFO ontology needs to be populated so it is not only an empty conceptual shell but a living part of the system. These topics will be further worked on in the project and will be as well documented in D2.3.2 and to a lesser extent in D2.2.2.

6 References

1. **T.Tran, P. Ciminiano, A. Ankolekar.** A Rule-based Adaptation Model for Ontology-based Personalization. *Studies in Computational Intelligence, Vol. 93/2008.* 2008.

2. *Semantic software engineering: the role and usage of ontologies in information systems.* **T. Tran, H. Lewen, P. Haase.** Proceedings of the 5th IEEE International Conference on Computer Science Research, Innovation and Vision for the Future, 2007.

3. **Motti, V. et al.** *D2.1.1: CARF and CADS (R1).* 2011.

4. *A unifying reference framework for multi-target user interfaces.* **Calvary G., Coutaz J., Thevenin D., Limbourg Q., Bouillon L., Vanderdonckt, J.** 3, 2003, Interacting With Computers, Vol. 15, págs. 289-308.

5. *Usability of Nomadic User Interfaces.* **Dees, W.** Springer Berlin, 2011, LNCS: Human-Computer Interaction. Towards Mobile and Intelligent Interaction Environments.

6. **Kifer, M.** World Wide Web Consortium. *RIF Overview.* [online] 2010. http://www.w3.org/TR/rif-overview/.

7. *XSPARQL: Traveling between the XML and RDF worlds – and avoiding the XSLT pilgrimage.* **Akhtar, W., Kopecky, J., Krennwallner, T., Polleres, A.** [ed.] M. Koubarakis, & S. Bechhofer M. Hauswirth. 2008. Vol. Proceedings of the 5th European Semantic Web Conference. Retrieved from http://data.semanticweb.org/conference/eswc/2008/papers/119.

8. **Prud'hommeauc, E., Seaborne, A.** *SPARQL Query Language for RDF -- W3C Recommendation.* World Wide Web Consortium, 2008.

9. *XSLT+SPARQL : Scripting the Semantic Web with SPARQL embedded into XSLT stylesheets.* **Berrueta, D., Labra, J. E., & Herman, I.** [ed.] S. Auer, G. A. Grimmes, & T. Heath Chris Bizer. 2008. 4th Workshop of Scripting for the Semantic Web.

10. **Patel-Schneider, P., Hayes, P., Horrocks, I.** OWL Web Ontology Language Abstract Syntax and Semantics -- W3C Recommendation. [online] World Wide Web Consortium, 2004. http://www.w3.org/TR/owl-semantics.

11. **Carroll, J.J., Roo, J. D.** OWL Web Ontology Language Test Cases - W3C Recommendation. [online] W3C, 2004. http://www.w3.org/TR/owl-test.

12. *Consistency Checking of Semantic Web Ontologies.* **Kenneth Baclawski, Mieczyslaw M. Kokar, Richard Waldinger, Paul A. Kogut.** Springer, 2002. First International Semantic Web Conference (ISWC 2002). págs. 454-459.

13. **OilEd.** OilEd. Ontology editor for DAML+OIL. [online] http://oiled.man.ac.uk.

14. **Horrocks, I.** FaCT: Fast Classification of Terminologies. [online] http://www.cs.man.ac.uk/horrocks/FaCT.

15. **Frank, G.** Hybrid reasoning architecture general purpose first-order logic theorem prover suite of special-purpose reasoners. [online] http://www.ksl.stanford.edu/software/JTP.

16. **U., Stanford.** Chimaera. [online] Stanford University. http://www.ksl.stanford.edu/software/chimaera.

17. *An environment for merging and testing large ontologies.* **D. McGuinness, R. Fikes, J. Rice, and S. Wilder.** Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000).

18. *Pellet: A practical OWL-DL reasoner.* **Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., Katz, Y.** 2007, Web Semantics: Science, Services and Agents on the World Wide Web In Software Engineering and the Semantic Web, Vol. 5, No. 2., págs. 51-53.

19. **Fonseca, J. M. C., & Lewis, R.** *Delivery Context Ontology -- W3C Working Draft.* World Wide Web Consortium, 2009.

20. *FOAF Vocabulary Specification 0.98.* **Brickley, D., Miller, L.** [ed.] D. Brickley & L. Miller. 2010.

Document 9 August 2010 Marco Polo Edition. Retrieved from http://xmlns.com/foaf/spec/.

21. *Towards Semantically-Interlinked Online Communities.* **Breslin, J. G., Harth, A., Bojars, U., & Decker, S.** [ed.] A. Gómez-Pérez & J. Euzenat. 2005, The Semantic Web Research and Applications, págs. 500-514.

22. **Klyne, G., Reynolds, F., Woodrow, C., Hidetaka, O., Hjelm, J., Butler, M. K., & Tran, L.** *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0.* World Wide Web Consortium, 2004.

23. *Semantic user preference descriptions in MPEG-7/21.* **Tsinaraki, C., & Christodoulakis, S.** 2005. Hellenistic Data Management Symposium HDMS.

24. *Foundations of preferences in database systems.* **Kießling, W.** Morgan Kaufmann, 2002. VLDB '02 Proceedings of the 28th international conference on Very Large Data Bases. págs. 311-322.

25. *CC/PP and UAProf: Issues, improvements and future directions.* **Butler, M H.** 2002. Proceedings of W3C Delivery Context Workshop DIWS 2002.

26. **Butler, Mark H.** *Some questions and answers on CC/PP and UAProf.* 2002. HP Technical Report. Retrieved http://www.hpl.hp.com/techreports/2002/HPL-2002-73.pdf.

27. *Experiences in Using CC/PP in Context-Aware Systems.* **Indulska, J., Robinson, R., Rakotonirainy, A., & Henricksen, K.** Mobile Data Management , págs. 247-261.

28. *Addressing On-Demand Assembly and Adaptation Using a Runtime Intentional Versioning Engine.* **Gergič, J.** 2008, Charles University in Prague.

29. *Linked Data: Evolving the Web into a Global Data Space.* **Heath, T., & Bizer, Christian.** [ed.] Eds. J. Hendler & F. V. Harmelen. Morgan & Claypool, 2011. Proceeding of the 17th international conference on World Wide Web WWW 08 . Vol. 1, pág. 1265.

30. *Ontology based context modeling and reasoning using OWL.* **Wang, X. H., Gu, T., & Pung, H. K.** [ed.] G. Tao, & P. Hung Keng Z. Da Qing. s.l. : IEEE. Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops 2004. págs. 18-22.

31. **Caminero, J. et al.** *SERENOA D1.2.1 - Architectural Specifications (R1).* 2011. FP7 SERENOA Project Deliverable.

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

- **TID:** Telefónica I+D (partner name)
- **UCL**: Université Catholique de Louvain (partner name)
- **ISTI**: Consiglio Nazionale delle Ricerche (partner name)
- **SAP**: SAP AG (partner name)
- **W3C**: Geie Ercim (partner name)
- **W4**: W4 (partner name)
- **CTIC**: Fundación CTIC – Centro tecnológico para el desarrollo en Asturias de las Tecnologías de la Información (partner name)

- **AAL:** Advanced Adaptation logic
- **Adaptability**: The capacity of a UI to adapt its behaviour through explicit human intervention.
- **Adaptable** User Interface: A UI that supports adaptability.
- **Adaptivity**: The capacity of a UI to adapt without any explicit human intervention.
- **API:** Application Programmers' Interface
- **AUI / Abstract User Interface**: A description of a UI that is independent from the specific UI resources available on the target computing platform.
- **CAA**: Context-aware Adaptation
- **CADS**: Context-Aware Design Space
- **CARF**: Context-Aware Reference Framework
- **CARFO**: CARF Ontology; an ontology that provides concepts for the representation of all knowledge in SERENOA's domain.
- **Context**: A context identifies the situation in which e.g. a certain action occurs. Several aspects can be considered within the context: the user, the device, the environment…
- **CUI / Concrete User Interface**: A description of a UI that is dependent on the specific UI resources and modalities available on the target computing platform.
- **DAML+OIL:** successor language to DAML and OIL that combines features of both. Superseded by Web Ontology Language (OWL).
- **DL / Description Logic**: A family of formal knowledge representation languages. It is more expressive than propositional logic but has more efficient decision problems than first-order predicate logic. It is used in artificial intelligence for formal reasoning on the concepts of an application domain (known as terminological knowledge). It is of particular importance in providing a logical formalism for ontologies and the Semantic Web.
- **First Order Logic**: Formal logical system used in mathematics, philosophy, linguistics, and computer science. Also referred to as first-order predicate calculus, the lower predicate calculus, quantification theory, and predicate logic. It is distinguished from propositional logic by its use of quantifiers.
- **FUI / Final User Interface**: The UI produced at the implementation level, expressed as source code.
- **GUI**: Graphical User Interface: "In computing a graphical user interface (GUI, sometimes pronounced gooey) is a type of user interface that allows users to interact with electronic devices

with images rather than text commands. … A GUI represents the information and actions available to a user through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation.” [7]

- **HCI**: Human Computer Interaction: “Human–computer interaction (HCI) is the study, planning and design of the interaction between people (users) and computers. It is often regarded as the intersection of computer science, behavioural sciences, design and several other fields of study.” [8]
- **ICT**: Information and Communication Technologies
- **IDE**: Integrated Development Environment: “An integrated development environment (IDE) also known as integrated design environment or integrated debugging environment is a software application that provides comprehensive facilities to computer programmers for software development.” [9]
- **Interactor**: A single interaction object
- **MDA**: Model-Driven Architecture: the population of the software development process with difference models, each representing a particular view on the system being built
- **MDE**: Model-Driven Engineering
- **MEP**: Member European Parliament
- **Mereology**: the theory of parthood relations such of the relations of part to whole and the relations of part to part within a whole
- **Meta-UI**: Interactive system whose set of functions is necessary and sufficient to control and evaluate the state of an interactive ambient space
- **Ontology**: Formal and explicit specification of known concepts
- **OWL**: Web Ontology Language, W3C standard for ontology representation
- **Platform**: A class of devices that share the same characteristics in terms of interaction resources. Examples of platforms are the graphical desktop, PDAs, mobile phones, vocal systems…
- **QoSFE**: Quality of Front-End Services
- **RDF:** Family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model. Used as a general method for conceptual description or modelling of information that is implemented in web resources, using a variety of syntax formats.
- **Re-Molding**: Exploiting different modalities
- **SFE**: Service Front-End: Service front-ends are interfaces between the user and the backend through which the user gets access to various services.
- **SOA**: Service-oriented Architecture, A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. [10]
- **UI**: User Interface: “In the industrial design field of human–machine interaction, the user interface is the space where interaction between humans and machines occurs. The goal of interaction between a

[7] Wikipedia online cited: 03 February 2011. http://en.wikipedia.org/wiki/Graphical_user_interface

[8] Wikipedia online cited: 03 February 2011. http://en.wikipedia.org/wiki/Human–computer_interaction

[9] Wikipedia online cited: 03 February 2011.http://en.wikipedia.org/wiki/Integrated_development_environment

[10] http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html

human and a machine at the user interface is effective operation and control of the machine, and feedback from the machine which aids the operator in making operational decisions."[11]

- **XQuery**: W3C sanctioned query and functional programming language that is designed to query collections of XML data.

[11] Wikipedia online cited: 03 February 2011.http://en.wikipedia.org/wiki/User_interface