A Context-Aware Semantic Approach for the Effective Selection of an Assistive Software

Andrés Iglesias-Pérez, Marino Linaje, Juan Carlos Preciado, Fernando Sánchez-Figueroa
Escuela Politécnica de Cáceres
Universidad de Extremadura
10003 Cáceres (Spain)
{andresip, mlinaje, jcpreciado, fernando}@unex.es

Elena Gómez-Martínez, Rafael González-Cabero
R&D Department,
Technosite,
Albasanz , 16
28037 Madrid (Spain)
{megomez,rgonzalez}@technosite.es

José Ángel Martínez-Usero
International Projects Department,
Technosite,
Square de Meeus, 37
Brussels (Belgium)
jamartinez@technosite.es

Abstract

People in industrial societies usually carry at least one electronic device (e.g., mobile, console) with some kind of connectivity support (e.g., Bluetooth, WI-FI). Although we know how to interact with these devices, it is still hard to interact with auto-discovered systems that are present in the environment (e.g., using different languages, etc.). This situation is even more problematic for people with functional limitations who need to interact with inaccessible and unknown user interfaces.

One of the main problems of the auto-discovery architectures for people with functional limitations is to find the most appropriate assistive software to be used in the user's device in order to interact with the auto-discovered system according to the context. This paper is focused on the domain knowledge conceptualization as well as the required mechanisms to infer new knowledge with the aim of providing the basis for the automatic selection of the assistive software.

By means of the system presented, the AS can be fully automatically selected according to the user profile, the user device capabilities and the AmI target device. For this purpose the interactions between these three actors was rethought from their basis to take into account people with functional limitations. The AS ontology has been specified using Methontology. This methodology is well known, follows an engineering approach and it fits better with collaborative works, such as INREDIS project, where many researches and enterprises worked together to fix the ontology presented here.

1. Introduction and motivation

The European market of Assistive Software (AS) is very fragmented and non-structured. Nevertheless, an enormous amount of information on AS is already available, but does not reach the end-users. Therefore, a centralised initiative to foster delivering of AS according to final user needs in different ICT environments is needed at European level. In the present research, a key issue to make this European initiative possible is introduced. With the aim of establishing a motivating use case, let's figure that a blind user enters a room with an air-conditioning system (AC). The usual case is that the AC offers only a visual user interface. Under this situation the user is unable to perform any task on it. The ambient should offer an alternative interface to interact with the AC, namely AS. This interface should be based on the context (e.g., the user profile or the device capabilities).

Context, in the scope of the proposal, includes the auto-discovered target devices potentially present in the environment, the user who wants to interact with them and the user's device that will be used as intermediary in the interaction process. The proposal assumes that a) these target devices are able to expose their user interfaces through a common architecture, b) the user's device capabilities can be fetched by the architecture and c) the user functional limitations are managed by the architecture. Under these inconstant conditions, our context-aware solution is able to select the most suitable AS for that interaction according to that given context following a semantic approach.

Even when the approach presented in this paper uses services of the INREDIS architecture, we underline that it has a general validity not confined to the use inside that project. So other auto-discovery architectures could take advantage of this research.

The paper is organized as follows. Section 2 presents the related work. Section 3 briefly summarizes the INREDIS architecture. Section 4 is focused on the AS ontology and the automatic selection system. Finally Section 5 outlines the conclusions and future work.

2. Related work

The selection of an AS has been traditionally based on trial and error, following these steps: 1) check a user interface and notice that it is inaccessible, 2) find and AS that claims to solve this interaction issue, 3) install it, 4) if the AS does not solve the interaction, go to step 1.

Selection procedure is problematic and the technology changes quickly. Therefore, it has to be performed quite often (e.g. a new version of web browser may require a new version of Text-To_Speech. This situation makes the user to spend money and time testing AS that finally will not effective solve the problem. Also, new paradigms to deploy and use AS have recently arisen (e.g., Software as a Service). These ways to select assistive software are unfeasible for the use case presented in Section 1, since they require a very well-planned interaction, not a casual one with new elements. To the best of our knowledge, there are no automated proposals which perform the task of selecting AS. However, a categorization of different approaches for the general purpose of storing assistive product information can be provided as follows.

On the one hand, there exist data-centred approaches. ISO 9999:2007[6] establishes a classification of assistive products, including AS. Based on this categorization, some organizations, such as the European EASTIN network, have developed databases in which collect a set of assistive products in the current marketplace. They try to guide the users and their relatives, and reduce the number of AS to try, but that help is currently out-of-the-computer, produced by a domain expert and written as a set of guidelines, catalogues and web pages. This issue restricts automatic selection aids while it is necessary to systematize them [1].

In this way, Masuwa-Morgan and Burrell[11] emphasize the need for an ontology to minimize accessibility requirement specifications. So, in [12] an ontology named AccessOnto is proposed. Nevertheless, this ontology neither considers the capabilities of a user nor his/her accessibility demands. Additionally it lacks of AS specification and categorization and, actually, being a data centred approach without reasoning process.

On the other hand, there exist Knowledge-centred approaches. Among them, Karim et al.[7] focus on an impaired user interface ontology for the content adaptation. Although their proposal generates user interfaces personalized by impairment, they only consider theoretical web interface content adaptation without a reasoning process. Concerning assisted living and assistive technologies, Klein et al.[8] present an exhaustive ontology-centred design to model context in smart homes based on AmI. Issues such as user's capabilities, AS or user's interactions are not contemplated.

Castro et al.[1] propose a repository of ontologies aimed at raising metadata interoperability across assistive technology open repositories by means of a vocabulary without considering the user's needs or ambient intelligence aspects.

Besides, ISO 24756:2009 [5]describes needs and capabilities in order to integrate assistive technologies, but they are only mentioned as external services.

Lessons learnt from the analysis of research somewhat close to AS selection are twofold. On the one hand, the trend is to favour taxonomies and ontologies instead of databases. On the other hand, there are two main approaches to solve the proposed use case: to delegate reasoning to a programmed ad-hoc access layer, or not to reason at all. No reasoning leads to no communication, so it is not a solution to the use case. Programmed reasoning may do useful matching between controller devices and assistive software, but it is prone to scatter the reasoning between several components of the architecture, making the interoperability more complicated to achieve. Our proposal fosters interoperability and maintainability by centring al the reasoning and the domain expertise on the same component.

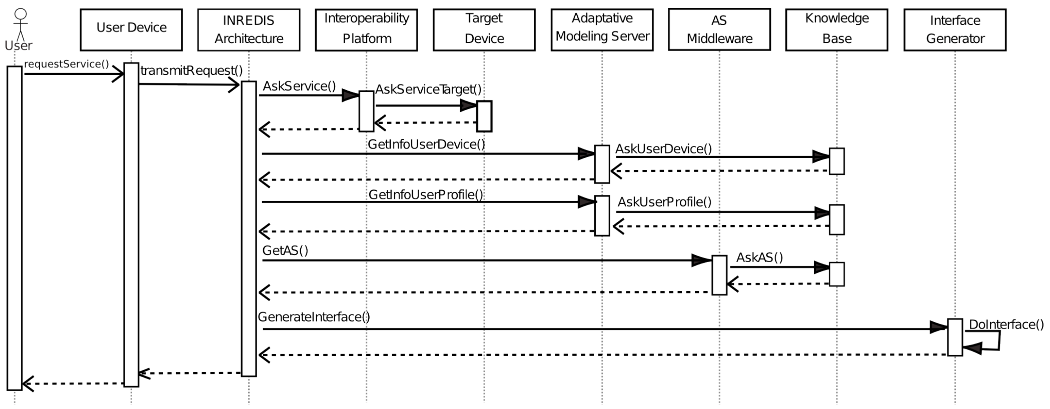

Figure 1. Sequence diagram describing the service request in INREDIS architecture for a SaaS AS.

3. INREDIS architecture

In this section we are going to outline the INREDIS architecture. It is an event-driven service-oriented architecture that further develops the idea of Universal Control Hub (UCH) [19]. The rationale of such approach is that a person with an adapted controller device (e.g. mobile phone) should be able to interact and control different devices by means of an interoperability architecture. Using a controller, the achievement of a proper control is eased significantly as the accessibility problem of the whole environment is reduced to just solving the accessibility issues with the user controller.

INREDIS architecture extends the concept of UCH providing new modules that enable activities of uttermost importance for people with special needs, such as interface adaptation, AS delivery, context and interaction adaptability. Even though we refer to them as a module of the architecture they are complex systems on their own, as the reader might have realized about the AS middleware that is the scope of this paper. The most important among these modules are:

- **Knowledge base**. The knowledge base stores all the ontologies that collect formal descriptions of the elements in the INREDIS domain (e.g. users, AS, devices, software requirements, etc.); and its instances. It provides mechanisms for reasoning with and querying all this knowledge, what enables the intelligent behaviour of other modules of the architecture.
- **Adaptive modeling server**. It is the module that keeps the knowledge base content updated. From different sources (application context, heterogeneous sources, user interaction logs, etc.) it creates new instances, concepts and terminological axioms of the knowledge base.
- **Assistive Software middleware**. This module provides ubiquitous access to a set of AS instances. It supports different AS deployment approaches (e.g., downloaded) but it uses a SaaS (Software as a Service) as the main paradigm approach.This component thus achieves on the one hand the provision of AS without prior installation; and on the other it brings the possibility of using AS that might surpass the computational resources capability of the user’s device.
- **Interface generator**. This module adapts interfaces expressed in a generic and abstract language into concrete, utilizable and accessible ones. This activity is made in terms of the user characteristics, the device capabilities and the context; and is made possible using the reasoning capabilities that the knowledge base provides.

From now on, terms in italics refer to terms that are conceptualized in the tables or figures provided in the paper.

The sequence diagram in Figure 1 models how a user requests a service from the *Target Device* assuming that user authentication process has been previously made. The *Target Device* has been chosen among a set of available auto-discovered target devices. The complex auto-discovery process is out of the scope of this paper. Firstly, the *User Device* communicates to INREDIS architecture core, which connects up *Interoperability Platform*, containing different interoperability protocols (such as UCH, OSGi and Web Services). The *Interoperability Platform* selects the corresponding protocol in order to interact to the specific *Target Device*. Once the service is known, it is necessary to adapt its user interface. So, the user's device information and the user's profile are asked for the *Adaptive Modeling Server*, which looks up the *Knowledge Base* about them. This information together with the context information is headed for *AS Middleware*, which queries to *Knowledge Base* the most appropriate *AS* and delivers it as a service, most concretely the mentioned Software as a Service approach. All collected information is passed to *Interface Generator* in order to provide the most suitable user interface.

From now, the paper is focused on the semantic approach for the effective selection of the most suitable AS. The core of this approach is an AS ontology within the middleware that is able to store AS-related knowledge, infer new knowledge and select the right AS for each case according to the context of use.

4. An Assistive Software Ontology and the selection system

The procedure of selecting an AS is not an easy task and it has to be performed many times and quite often. The solution presented here supports the automatic selection of the most appropriate AS according to the user profile (when available), the device capabilities and the AmI (Ambient Intelligence) target device to interact with. This automatic selection gives support to the integration of AmI target devices in everyday life of people with functional limitations using the most suitable AS to interact with them when required (e.g. to set the preferences when the user is detected in the room).

The proposal presented here is based on the classic Shannon schematic diagram of a general communication system [14] that has been adapted and extended in the context of our research. The adapted diagram is depicted in Figure 2. It must be taken into account that between two different actors of the communication (note that the UML actor icon in Figure 2 is not limited to represent people), several elements should be considered. However, Figure 2 is only focused on those ones that present situations where a user may experience inaccessible or handicapped interfaces.

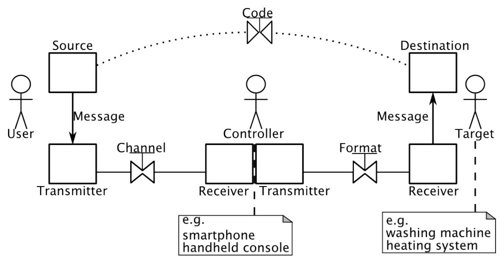

Figure 2. Schematic diagram of the approach

These potential handicapped interactions are depicted in Figure 2 by means of two confronted arrows separated by a vertical and a horizontal line (called valve) according to ISO [5]. The more closed the valve is, the more handicapping the situation is.

The main extension to the Shannon's proposal is that in the proposed schema not only those interfaces where a message is transmitted are specified but also those ones where there is not message transmission (marked with a dotted line in Figure 2). With this extension, handicapping situations between the *User* and the *Target* due to the *Code* that each one uses can be treated. A typical example of *Code* handicapped interaction is e.g., a *Target* heating system with audio controls in German that cannot be managed by *Users* who only speak Japanese. To represent this case, we fully closed the *Code* valve in Figure 2.

The other extension is the inclusion of *Channel* and *Format*. Here, in the communication between the *User* and the *Controller* only the *Channel* is relevant. Also, between the *Controller* and the *Target* only the *Format* is relevant to conceptualize handicapped interactions. All the concepts specified in Figure 2 that has correspondence with [14] are already defined there. The rest of the relevant concepts will be formally conceptualized and explained in Section 3.1.

Among the different technologies to represent, store and infer new knowledge from the system, we have used an ontology. This decision is based on previous research e.g., [11]and experiences of some of the authors e.g., [10] that demonstrated that ontologies are a good approach to support accessibility issues. Additionally, the use of a semantic approach makes the proposal here presented easier to mix with other auto-discovery architectures in the future. According to [16], an ontology is a formal, explicit specification of a shared conceptualization. This definition fits perfect with the cooperative process followed by the presented work in the context of the INREDIS project.

An ontology must be formalized and implemented in order to be processed by a system that automatically infers new concept for it. However, like many other complex engineering processes, ontologies can be specified following many different methodologies. Due to the amount of people collaborating and reviewing this research work, we decided to use Methontology [2]. Methontology allows the cooperative specification of the ontology using a refinement process until the authors and reviewers reach an agreement.

The different activities of Methontology are Specification, Conceptualization, Formalization, Implementation and Maintenance. However, not all the activities are relevant for this research paper. While Specification has been already stated in Section 1 (i.e., motivation), Conceptualization is covered in subsection 3.1. Different alternatives are available for Formalization and Implementation and it has been an implementation decision (out of the scope of this paper) to use OWL [17]. This decision was based on the need to process the content of information instead of just presenting information to humans and the full set of tools available for it. Finally, Maintenance is out of the scope of the paper, since it does not imply any relevant research.

4.1. Conceptualization of Assistive Software

The conceptualization activity organizes and converts an informally perceived view of a domain into a semi-formal specification using a set of intermediate representations. For this purpose, different tasks must be defined.

Due to the length of the documentation that we obtained following Methontology and the paper limited size, we do not go into details for all the tasks in this section. Notwithstanding, an excerpt of each task will be given (or relevant comments/guidelines when necessary).

For all the cooperative tasks during the conceptualization activity, each task goal has been specified using a versioned online spreadsheet and a wiki to discuss them through an iterative refinement process.

It is a fact that ontology conceptualization is done from a subjective point of view since it is performed by persons. However, even when the reader may think in another way to conceptualize the terms, relations, etc., we should think that an ontology must be a consensus for its wide-adoption and this research represents a consensus of many people from different scientific and business areas. It is important to note that the proposal here presented makes use of standards when possible (e.g., Task 6 provides an example).

Name	Synonym	Acronym	Description	Type
Authentication Method	Authentication Mechanism	AM	User's way to access restricted or customized contents in the AS.	Concept
User Controller Interface		U_C_I	The point of interaction between the User and the Device.	Concept
Code			Set of features that the message has to comply with in order to be understood by Source and Destination	Concept
Format			Logical coding standard that Devices use to transmit the information	Concept

Table 1. Excerpt of Task 1. Glossary of terms.

Concept Name	Instances	Class Attributes	Instance attributes	Relations
Authentication Method	notNecessary; byCookie byPrompt; byCaptcha	authenticationPlace	authenticationType	-
User Controller Interface	Screen1; Qwerty4	Source Destination	transmitter receiver	hasChannel hasCapableTransmitter

Table 2. Excerpt of Task 4. Concept dictionary.

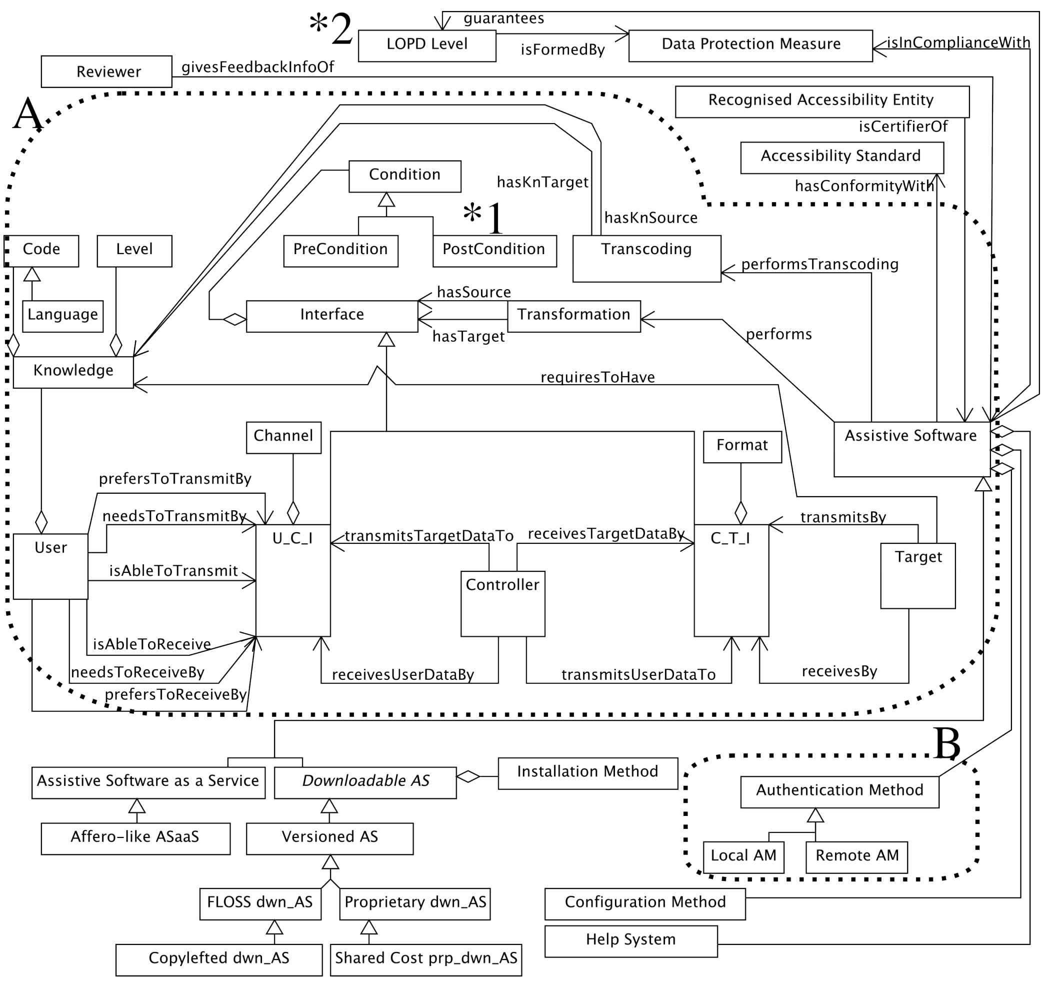

Figure 3. Task 2 and 3. Concept taxonomies and ad-hoc binary relations diagrams

Relation Name	Source concept	Cardinality	Target Concept	Inverse Relations
hasCapableTransmitter	U_C_I	N	User	isAbleToTransmit
hasCapableReceiver	User	N	U_C_I	isAbleToReceive
isAMof	AM	N	Assistive Software	hasAM

Table 3. Excerpt of Task 5. Detailed Ad-hoc binary relations.

Instance Attribute Name	Concept name	Value type	Value range	Cardinality
authenticationType	AM	String	--	(1,1)
Transmitter	U_C_I	String	[UC]	(1,2)
Receiver	U_C_I	String	[UC]	(1,2)
RFC5646 langTuple	Language	String	ISO 639-3	(1,N)

Table 4. Excerpt of Task 6. Detailed instance attributes.

Class attribute Name	Defined concept	Value type	Cardinality	Values
authenticationPlace	Local AM	[local,remote]	(1,2)	Local
authenticationPlace	Remote AM	[local,remote]	(1,2)	Remote
Source	U_C_I	[User, Target]	(1,1)	User
Source	C_T_I	[User, Target]	(1,1)	Target

Table 5. Excerpt of Task 7. Detailed class attributes.

Axiom name	Description	Expression	Referred concepts	Referred relations	Vars.
SaaS with Remote AM	SaaS Assistive Software must have remote authentication methods	$(\forall pa \ni Assistive\_Software(pa))$ $(\forall am \ni Authentication\_Method(am))$ $\neg(hasAM(?pa,?am) \wedge$ $Assistive\_Software\_as\_a\_Service(?pa)$ $\wedge \neg RemoteAM(?am))$	AS AM Assistive SaaS Remote AM	hasAM	?pa ?am
Downloaded with Local AM	Downloaded Assistive Software must have local Authentication Methods	$(\forall pa \ni Assistive\_Software(pa))$ $(\forall am \ni Authentication\_Method(am))$ $\neg(hasAM(?pa,?am) \wedge$ $Downloadable\_AS(?pa)$ $\wedge \neg LocalAM(?am))$	AS AM Downloadable_AS LocalAM	hasAM	?pa ?am

Table 6. Excerpt of Task 9. Formal axioms.

Rule name	Description	Expression	Concepts	Referred relations	Vars.
AS suitable for a User Controller Interface	Given an User and a Controller Device, assert the Assistive Software that can be used to ensure the correct interface between them.	$User(?us) \wedge$ $isAbleToReceive(?us,?uci1) \wedge$ $Controller(?cd) \wedge$ $transmitsTargetDataTo(?cd,?uci2) \wedge$ $Transformation(?tf) \wedge$ $hasSource(?tf,?uci2) \wedge$ $hasTarget(?tf,?uci1) \wedge$ $performs(?as,?tf) \wedge$ $swrlx:makeOWLThing(?ai,?as)$ $\Rightarrow$ $AssistedInterface(?ai) \wedge$ $isFeasibleFor(?as,?ai) \wedge$ $hasRawTransmitter(?ai,?cd) \wedge$ $hasAssistedReceiver(?ai,?us)$	User Controller, Transform. ASforUCI, (AssistedInterface)	isAbleToReceive transmitsTargetDataTo hasSource hasTarget performs	?us ?cd ?as ?uci1 ?uci2 ?tf ?ai

Table 7. Excerpt of Task 10. Rules.

Task 1. A glossary of terms is the output of this task. An excerpt of the full table is showed in Table 1 where two terms are explained (i.e., *User Controller Interface* and *Authentication Method*). These two terms will be used along this paper as excerpts for the rest of the tasks when possible. All the communication elements of Figure 2 are defined in the ontology. Those ones that are not defined in [14] (i.e., *Code* and *Format*) are also included in the excerpt of Table 1. This task is a valuable element to ensure that every person implied in the AS ontology conceptualization process is thinking in the same concept when a term is used.

Task 2 and 3. These tasks are used to specify the concept taxonomies and the ad-hoc binary relations diagram. Although Methontology does not fix the type of diagram to apply, we use here the standard UML (Unified Modeling Language) [13] class diagram because of its formality, conciseness and scalability. Tasks 2 and 3 outputs are summarized in Figure 3. As an example, let us consider the *Interface* class which is specialized in two *Interfaces*, one between *User* and *Controller* (i.e., *U_C_I*) and another between *Controller* and *Target* (i.e., *C_T_I*). That allows to share common properties to the value partitions [3] of *Channel* and *Format*, as well as to express richer transformations from one *Interface* to another. E.g., transforming a "normal" visual *Channel* to another visual *Channel* with screen magnification that transforms a specific *U_C_I* with *PostCondition* (marked *1 on Figure 3) 250%.

Task 4. This task output is the concept dictionary where the properties and relations that describe each concept of the taxonomy are specified. An excerpt of the full table is showed in Table 2.

It can be observed that *U_C_I* has less relations than expected according to Figure 3 (i.e., *hasChannel* and *hasCapableTransmitter*). This is due to the fact that in this table only the relations in which *U_C_I* acts as source of the relation are stated. That is also the reason why *AM* has no relations on this table while the rest of the columns (i.e., instances, class attributes and instance attributes) are filled for both of them.

Task 5. It has the primary goal of describing in detail all the ad hoc binary relations included in the concept dictionary. An excerpt of the full table is showed in Table 3.

For example, for *U_C_I* the relation *hasCapableTransmitter* is specified. Of course, many other relations have been identified. Although here we do not fully follows Methontology and due to their importance for the reader to fully understand this ontology, the most important relations have been added to Figure 3 that really stands for Task 2 and 3. In the full documentation of the ontology we used both representations (i.e., UML and table).

Task 6. It is focused on the definition of instance attributes in detail. An excerpt of this task is showed in Table 4.

Here, *Transmitter* and *Receiver* attributes are specified in order to identify which communication elements are acting with these roles. That supports *Interfaces* that can be used to transmit and receive by both i.e., the *User* and the *Controller*.

According to Task 2, *Language* is a subclass of *Code*, conceptualized in our ontology by means of the tuple suggested by RFC 5646 and using ISO 639-3 tags for language. Though the expected scope of the use of our ontology currently indicates that only *language* and *region* tags will be necessary (for the current version of INREDIS architecture), we decided to provide support for the standard "language- extlang- script- region- variants– extensions- privateuse" tuple. This was decided in order to obtain a more general ontology to facilitate INREDIS architecture extensions as well as to bridge with other architectures in the future.

Task 7. The main goal to be achieved with this task is to specify in detail all the class attributes of the concept dictionary. Table 5 shows an excerpt of the full class attribute table.

Class attributes are used to make the classification of concepts under their sub-taxonomies. E.g., whether an *AM* can be considered as "belonging" to *Local AM* or *Remote AM* subclasses can be set up by means of asserting its *authenticationPlace*. A method can be both local and remote, so the attribute may be present twice for a given concept.

Task 8. It is the task devoted to the detailed specification of each constant defined in the glossary of terms. Due to the fact that the glossary of terms of this ontology does not require the specification of any constant, it makes no sense to complete this task.

Task 9. It is the task of the method dedicated to define formal axioms. In tasks 3 engineers are not capable to express the richness of the potential relations between the concepts, so it is necessary to outline first-order logic sentences that conceptualize them by means of constrains.

Table 6 represents an excerpt of the full set of axioms that are specified in the ontology. This excerpt shows the axioms to support the relation between *AS* and *AM*. In the first row of the table it is specified that a *SaaS AS* must have all its *Authentication Methods* fixed as *Remote* (i.e., *AM* asserted as *Remote AM*). In the second row it is specified that a *Downloadable_AS* must have all its *Authentication Method* fixed as *Local.*

Task 10. This task is used to specify which rules are needed. More explicit rules make querying easier and faster, at the cost of increasing the amount of assertions. However, at this point it is a design decision which is still independent of the code and, therefore, it can be easily balanced later on according to the deployment facilities.

Table 7 represents an excerpt of the output of this task. In the excerpt is shown the rule to assert that an *AS* can be used to ensure a correct interface between the *User* and the *Controller*, that plays a pivotal role from the *AS* point of view. An extension detailed in [15] and the SWRL standard [18] have been used in order to create individuals. Since this fact is bounded to a specific solution rather to the specification of the problem, the concept written between gaps (i.e., *AssistedInterface*) and its relations have not been specified as part of Tasks 2 and 3. We highlight that it has been introduced a special *AS* named "notNecessary" that simulates a null *Transformation* between an *Interface* and the same *Interface.* It is necessary for the case of non-handicapped interaction and it eliminates the ambiguity between "Assistive Software Not Necessary" and "Assistive Software Not Found" cases.

Task 11. The last task in the method is used to define instances. We do not cover this issue here due to the low research contribution of this point. However, we outline a general interesting guideline to include them.

The use of our ontology leads the user profile to be specified using a set of checkmarks stating the user capabilities, needs, preferences and knowledge. As an example, for a "legally blind" user is not asserted the property "canReceive" for any *Interface* with the Visual *Channel*. This also makes this proposal easier to comply with the current regulations and laws of different countries. For demonstration purposes and due to the fact that the system is held in Spain, we added *LOPD* (marked *2 at the top-right corner in Figure 3), a legal data protection act. Other regulations and laws could be added to the ontology following the same pattern, since the *Data Protection Measure* class is independent of them.

4.2. Selection of the most appropriated Assistive Software

For cases where more than an *AS* is asserted to be suitable to solve the handicapped interaction, it is necessary to establish an order of preference. This is done based on a secondary subset of the concepts presented in the ontology. The number of *AS* products that match the interaction requirements is growing and users are asking for automatic mechanisms under many situations. Therefore, it is necessary to provide the ontology with mechanisms for the automatic selection not only for a suitable set of *AS* for a given interaction, but also for the most appropriated one. The proposal presented here achieves this automation following the next two phases.

Firstly, the reasoning pre-selects all the suitable *AS* candidates to be used in the interaction according to their semantic specification, using the set named "A" on Figure 3. Secondly, the rest of the concepts of the ontology are used to score every candidate, using the set ¬A Of course, that scoring is customizable and it can be performed more accurately if the user set his preferences on the auto-discovery platform (e.g., the INREDIS one). However, this is not a barrier for privacy concerns [9] and there is a default set of values to support anonymous users.

Finally, there are two methods to present the user with the most appropriated *AS* through the controller device in use. In the first one, a transparent selection from the user point of view is carried out and following this method, a fully automatic approach is achieved. To use this method, *Users* must state on their profiles that the best scored *AS* must be used without notification ("I feel lucky" option that it is activated by default). In the second method, a semi-automatic *AS* selection is carried out. This kind of selection is typically triggered automatically by the system when the user refuses the automatically provided *AS* following the first method. In this case, an ordered by the scoring list is sent to the user as a suggestion and it is the user who finally selects the desired *AS*.

5. Future Work

An ontology to store knowledge, to infer new knowledge and to support the automatic selection of the most suitable AS for a given interaction context has been presented.

However, there is already future work to carry out to increase current functionality. Actually, neither INREDIS architecture nor ontology provide the required mechanisms to support user profile feedback. This knowledge could be exploited using machine learning techniques in order to provide additional artificial intelligence to the system to e.g., provide automatic user profile updates according to the AS selections that the user carries out after several interactions with the system. Also, as compelling proposals on the subject arise, it will be performed an effort in order to evaluate the full set, including this one presented here.

6. Acknowledgements

The research described in this paper arises from a Spanish research project called INREDIS (INterfaces for RElations between Environment and people with DISabilities) [4]. INREDIS is led by Technosite and funded by CDTI (Industrial Technology Development Centre), under the CENIT (National Strategic Technical Research Consortia) Programme, in the framework of the Spanish government's INGENIO 2010 initiative.

The opinions expressed in this paper are those of the authors and are not necessarily those of the INREDIS project's partners or of the CDTI.

Thanks to all the reviewers of the project that acted as reviewers for the ontology development.

References

[1] Castro, A.G. Normann, I. Hois, J. Kutz, O. Ontologizing Metadata for Assistive Technologies - The OASIS Repository., International Workshop on Ontologies in Interactive Systems, pp. 57-62, 2008.

[2] Corcho, O., Fernández-López, M., Gómez-Pérez, A. Methodologies tools and languages for building ontologies. Where is their meeting point?. Data & Knowledge Engineering, vol 46, iss. 1, pp. 41-46, 2003.

[3] Horridge, M., Knublauch, H., Rector, A, Stevens, R., Wroe, C. A practical guide to building OWL ontologies using the Protégé-OWL plugin and CO-ODE tools. University of Manchester, 2004.

[4] INREDIS project. http://www.inredis.es

[5] ISO JTC 1 - SC 35. 24756:2009. Framework for specifying a common access profile (CAP) of needs and capabilities of users, systems, and their environments, 2009.

[6] ISO TC 173 - SC 2. 9999:2007. Assistive products for persons with disability -- Classification and terminology, 2007.

[7] Karim, S., Tjoa, A. M. Towards the Use of Ontologies for Improving User Interaction for People with Special Needs. International Conference Computers Helping People with Special Needs, LNCS vol. 4061, pp. 77-84, 2006.

[8] Klein, M., Schmidt, A., Lauer, R. Ontology-Centred Design of an Ambient Middleware for Assisted Living: The Case of SOPRANO. Towards Ambient Intelligence: Methods for Cooperating Ensembles in Ubiquitous Environments, 30th Annual German Conference on Artificial Intelligence, LNAI vol. 4667, 2007.

[9] Kobsa, A. Privacy-enhanced personalization. Communications of the ACM, vol. 50, iss. 8, pp 24-33, 2007.

[10] Linaje, M., Lozano-Tello, A., Preciado, J.C., Sanchez-Figueroa, F., Rodriguez, R. Obtaining accessible RIA UIs by combining RUX-Method and SAW. Automated Specification and Verification of Web Systems, pp 85-98, 2009.

[11] Masuwa-Morgan, K.R., Burrell, P. Justification of the need for an ontology for accessibility requirements (theoretic framework). Interacting with Computers, vol. 16, iss. 13, pp. 523-555, 2004.

[12] Masuwa-Morgan K. Introducing AccessOnto: Ontology for Accessibility Requirements Specification. International Workshop on Ontologies in Interactive Systems, pp. 33-38, 2008.

[13] Rumbaugh, J., Jacobson, I., Booch, G. Unified Modeling Language reference Manual, the (2nd Edition). Pearson Higher Education, 2004.

[14] Shannon, C.E. A mathematical theory of communication. The Bell System Technical Journal, vol. 27, pp. 379–423, 623–656, 1948.

[15] Stanford University. Protégé extensions built-ins library http://protege.cim3.net/cgibin/wiki.pl?SWRLExtensionsBuiltIns , 2007

[16] Studer R., Benjamins V.R., Fensel D. Knowledge Engineering: Principles and Methods. IEEE Transactions on Data and Knowledge Engineering, vol. 25, iss. 1, pp. 161-197, 1998.

[17] W3C. OWL Web Ontology Language Overview, http://www.w3.org/TR/2004/REC-owl-features-20040210/, 2004.

[18] W3C. SWRL: A Semantic Web Rule Language. Combining OWL and RuleML http://www.w3.org/Submission/SWRL/ , 2004

[19] Zimmermann G., Vanderheiden G. The Universal Control Hub: An Open Platform for Remote User Interfaces in the Digital Home. Human-Computer Interaction. Interaction Platforms and Techniques, LNCS, vol. 4551, pp. 1040-1049, 200