A Semantic Approach for Designing Assistive Software Recommender Systems

Elena Gómez-Martínez[a], Marino Linaje[b], Fernando Sánchez-Figueroa[b], Andrés Iglesias-Pérez[e], Juan Carlos Preciado[b], Rafael González-Cabero[c], José Merseguer[d]

[a]*Babel Group. E.T.S. Ingenieros Informáticos, Universidad Pol. de Madrid, Spain*
[b]*Escuela Politécnica de Cáceres, Universidad de Extremadura, Spain*
[c]*Ontology Engineering Group. E.T.S. Ing. Informáticos, Universidad Pol. de Madrid, Spain*
[d]*Dpto. de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Spain*
[e]*Ilunion Accesibilidad Estudios y Proyectos, Spain*

Abstract

Assistive Software offers a solution for people with disabilities to manage specialized hardware, devices or services. However, these users may have difficulties in selecting and installing Assistive Software in their devices for managing smart environments. This paper addresses the requirements of these kinds of systems and their design in the context of interoperability architectures. Our solution follows a semantic approach, for which ontologies are a key. The paper also presents an implementation of our design proposal, i.e., a real and usable system which is evaluated according to a set of functional and non-functional requirements here proposed.

Keywords: Software design, Assistive software, Ontologies, Software non-functional evaluation

1. Introduction

Universal Access continues to be a critical quality target for Information and Communication Technologies (ICTs), as Stephanidis [1] stated. This is especially

Email addresses: `elena.gomez.martinez@upm.es` (Elena Gómez-Martínez), `mlinaje@unex.es` (Marino Linaje), `fernando@unex.es` (Fernando Sánchez-Figueroa), `andresip@unex.es` (Andrés Iglesias-Pérez), `jcpreciado@unex.es` (Juan Carlos Preciado), `rgonza@fi.upm.es` (Rafael González-Cabero), `jmerse@unizar.es` (José Merseguer)

important in industrial societies where there is a growing number of people with special needs[1], including those with aging-related conditions. Indeed, ICTs may require particular skills and abilities to interact with platforms, wireless communication systems and smart devices such as kiosks or ATMs.

Developing universally accessible smart environments is hard in terms of effort and required knowledge [3]. As an alternative, Assistive Software (AS from now on) provides an easy and feasible solution. AS represents software products specifically designed for people with some disability that is used to increase their ability to manage information in a digital device. AS therefore makes it easier to use ICT devices. This paper is mainly devoted to smart environments, e.g., the smart home [4]. For example, a blind person could use AS installed in her/his smartphone for managing mainstream software to control a smart TV or an air-conditioning system.

AS products can be selected in different ways. For example, simply using trial and error by 1) examining a user interface to determine whether it is accessible or not for a given disability (e.g., blindness), 2) finding an AS product that claims to solve the particular interaction issue, e.g., exploring assistive technology repositories such as EASTIN[2], 3) installing it, 4) returning to step 2 if the AS does not solve the interaction issue and so on. Using this manual form of AS selection, the user spends time and money testing AS products that in the end may not effectively solve the problem. Another interesting possibility is the use of assessment services [5, 6]. However, difficulties may arise in finding an Assistive Technology professional, e.g, in the very moment of browsing for finding the AS product. To address these issues, AS Recommender systems (ASR systems from now on) have been developed to help users in making decisions automatically and timely. An ASR system selects the most suitable AS for a specific context using as inputs the needs and preferences of the user, such as privacy, type of device used or type of disability.

This paper deals with the design of ASR systems and the requirements they should address. The design solution presented here is able to select the most suitable AS automatically, following a semantic approach. Indeed, the paper presents the conceptualization of an ontology for AS selection. Following the design guidelines, we implemented a Knowledge Base for the ontology and a real

[1]The terminology used in the paper as regards the disability field conforms to the International Classification of Functioning, Disability and Health [2].

[2]http://www.eastin.eu/

and usable ASR system, which is presented in the paper. The system successfully deals with non functional requirements such as response time and scalability.

To cope with complex environment necessities, ASR systems can be deployed in existing interoperability architectures. Such architectures allow access to heterogeneous data sources through semantically enriched services, e.g. using ontologies. For example, the SAPHIRE [7] interoperability architecture accesses disparate data sources to retrieve patient-specific information through Web services using standard medical ontologies. In our case, the architecture ensures that a) the user can interact with a controller device[3], b) the target devices or services can publish their user interfaces, c) the controller device capabilities can be shown and d) the disability of the user is managed by the architecture. The last two conditions are mandatory for a fully automatic ASR system, but optional for a semi-automatic selection. The ASR system presented here has been deployed in the context of the INREDIS [8] (INterfaces for RElations between Environment and people with DISabilities) interoperability architecture where it has also been evaluated.

The paper is organized as follows. Section 2 establishes the requirements for an ASR system. Sections 3 and 4 lay the design foundations for a semantic ASR system. Section 5 presents the ASR system we have developed and explains how it was deployed in INREDIS. Section 6 evaluates the ASR system in real scenarios. Section 7 describes related works. Finally, Section 8 outlines the conclusions and future work.

2. System Requirements

Tom has an impairment; specifically he is a blind user. He feels comfortable and safer carrying his smartphone, especially when he is out of his native Spain. After a long trip Tom has just arrived at the hotel in Tokyo where he has a reservation. It is his first time in this hotel. He enters the hotel room and his smartphone vibrates. It shows him some devices to interact with, such as the TV or the air-conditioning system (AC). He feels a little bit hot, so he will need to operate the AC through his smartphone to adjust the temperature and eventually to switch off the timer, for instance an hour later. This is an application scenario for an Assistive Software Recommender (ASR) system. Tom needs to automatically set up interfaces in his smartphone to operate the appliances in the hotel room.

[3]A *controller device* (e.g., a smartphone) allows the use of assistive technologies to bridge the gap between a user with a disability and a service or target device, e.g., a TV.

The requirements for ASR systems were carefully studied in the INREDIS [8] project. At the beginning of the project more than one thousand end-users were asked through questionnaires and interviews to list their needs and preferences. Specifically, 400 telephone surveys and 597 online surveys were carried out (257 with deaf persons and 340 with people with other disabilities). The respondents were randomly selected among ONCE[4] members. Moreover, seven discussion groups were set up for people with different impairments. In addition, fifteen open interviews were carried out with professionals, as "key informants", on different profiles of disability. All this information was useful for developing an initial prototype for self-detecting user's needs and capabilities according to ISO standard 24756:2009 [9]. Our conclusion is that an ASR system should meet the following requirements:

Requirement	*Description*
FR1	Detect "accessibility issues" for users with disabilities
FR2	Support anonymous and profile-based requests
FR3	Provide a weighted list of Assistive Software products automatically
FR4	Advise the user whether the selected AS is compliant with available data protection laws
FR5	Incorporate self-learning capabilities according to users' criteria
FR6	Install the selected Assistive Software automatically
FR7	Adapt to the needs of the user
NFR1	Be flexible, easy to use and communicative according to Nielsen principles [10]
NFR2	Have a response time according to Nielsen principles [10]
NFR3	Be scalable in well-defined environments (e.g., smart home and facilities)

Table 1: System requirements

Functional requirements (FR) define the scope of the system from the user point of view. The user will select the AS of her/his choice from a list (FR3, FR6). This list is built considering the needs and preferences of the user (FR1), who could also make requests to the system anonymously through generic profiles (FR2). FR5 confers learning capabilities on the system, from previous user selections.

Non-functional requirements (NFR) shape the system operation mode. In this

[4]ONCE is the main Spanish organization for blind people. It was a partner in the INREDIS project through one of its companies, Technosite.

respect, the main concern is to create an appropriate environment in which people with disabilities and elderly people can operate. In the aforementioned interviews, ONCE experts advised us that the Nielsen principles (NFR1 and NFR2) were a major source of satisfaction. Regarding system response time (NFR2), Nielsen establishes that:

- 0.1 second is about the limit for having the user feel that the system is reacting instantaneously.
- 1.0 second is about the limit for the user's flow of thought to stay uninterrupted, even though the user will notice the delay.
- 10 seconds is about the limit for keeping the user's attention focused on the dialogue. Users should be given feedback indicating when the computer expects to be done.

Although the target audience in our system has special needs, the response times must be similar to those for users without these needs not taking into account the time spent by disabled people in operating the target device[5]. Then, for the ASR system the expected response times should be within these intervals. Regarding scalability (NFR3), an ASR system should scale within the architecture where it is integrated, in our case the INREDIS architecture. INREDIS was projected for a wide range of real world scenarios with a high number of users. Examples of scenarios where INREDIS has been deployed are leisure services (location and purchasing tickets for events), smart homes [11], urban networking [12], social networks [13] or banking services (ATMs) [14].

Our functional requirements can be addressed at design level through several processes of knowledge management, which are described in Section 4. These processes rely on an ontology, described in Section 3. Table 2 matches each requirement with the process or processes that address it and the part of the ontology required. An assessment of the functional and non functional requirements is reported in Section 6.

3. An Ontology for ASR Systems

An ASR system implements processes, as described in Section 4, for accessing multiple sources of knowledge, most of them on the Web. For example, the

[5]For example, blind people interact with tactile interfaces by means of an immediate audible feedback.

Table 2: How system requirements are addressed through the paper

Requirement	*Processes in Section 4*	*Part of the ontology in Figure 1*
FR1	Detect Discrepancies and Check Feasibility	A
FR2	(1)	(2)
FR3	Match by Score	B,D,E,F
FR4	Score Privacy	C
FR5	Match by Log	(2)
FR6	Sort and Install	(2)
FR7	(3)	A
NFR1	Assessed in Section 6	
NFR2	Assessed in Section 6	
NFR3	Assessed in Section 6	

(1) The system can be anonymously accessed through an interface by default.
(2) The ontology does not apply here.
(3) The Interface Generator component, see Section 5.2, addresses this requirement by adapting the interface to the needs of the user.

Assistive Software products themselves[6] are deployed on the Web. Of the different technologies available for representing, storing and inferring knowledge, ontologies [15] have proved feasible for common semantic issues and especially for developing Web systems. Ontologies provide semantic-based advantages for systems that address accessibility requirements [16]. As a final argument, a semantic approach also facilitates the deployment of recommender systems within auto-discovery architectures, as we illustrate in Section 5.

According to [15], an ontology is a formal specification of a shared conceptualization. This definition fits with the cooperative process we have carried out to develop the AS ontology which was periodically reviewed by more than twenty researchers from different areas. The ontology here proposed addresses AS features and relationships, and its design ensures that all the information needed by the processes to cope with the system requirements is made available, as sum-

[6]According to EASTIN (www.eastin.eu), the principal Assistive Technology Information Network in Europe, the number of Assistive Products available in the EU increased to more than eighty thousand in 2014.

marized in Table 2. In order to be processed by a system, an ontology must be formalized and implemented; in this regard we have used the Methontology methodology [17].

Methontology allows cooperative specifications to be created using a refinement process until an agreement is reached and proposes a set of actions which we followed in order to achieve the conceptualization detailed below. We used the Web Ontology Language [18] for implementation, a widely adopted standard for this purpose. Other processes such as formalization and maintenance do not represent any relevant contribution to the goal of the paper, so they are not described here.

3.1. Ontology Conceptualization

To conceptualize means to organize and convert an informally perceived view of a domain into a semi-formal specification using different representations. For this purpose different cooperative tasks need to be defined. We specified each task using a versioned online spreadsheet and a wiki to describe them through an iterative review/refinement process. It may be considered that an ontology conceptualization is always done from a subjective viewpoint since it is performed by human beings. However, even if the terms and relationships presented here could be conceptualized in a different way, we must consider that an ontology requires a consensus for its widespread adoption. This research represents a consensus of many people from different scientific and business areas. Specifically, around 150 PhDs and engineers participated in the consensus. Around one thousand end-users were interviewed in the context of the INREDIS project. Some of these interviews were used as inputs for the ontology, for example to conceptualize communication channels. Psychologists and sociologists participated as usability and accessibility experts, also to conceptualize language related concepts. Electrical and electronic engineers advised on devices and controllers, while computer scientists and software engineers addressed mainly architecture-related issues. The project was assessed by 25 assistive technology experts. It is also important to note that our proposal makes use of standards to avoid subjective viewpoints whenever possible.

In the following we present the ontology through some relevant representations obtained during the conceptualization process.

Glossary of terms. One of the first documents needed for the conceptualization is a glossary that ensures that every person involved in the process understands the same concept when a term is used. The full output documentation contains around forty terms. Table 3 shows an explanation of five terms as an example.

Table 3: Glossary of terms

Name	Type	Synonym	Acronym	Description
Authentication Method	Concept	Authentication Mechanism	AM	User's way to access restricted or customized contents in the AS
User Controller Interface	Concept		U_C_I	The interaction point between the User and the Device
Code	Concept			Set of features that the message has to comply with to be understood by Source and Destination
Format	Concept			Logical coding standard that Devices use to transmit the information
Language	Concept			Languages that the AS supports: ISO i18n and localization

Taxonomy and ad-hoc binary relations diagram. Although Methontology does not prescribe the type of diagram to be applied, we have chosen the standard UML class diagram for several reasons: 1) it is independent of the final implementation; 2) it is sufficiently concise but has a broad set of native relationship types; 3) it is flexible and scalable when extensions are required.

Figure 1 illustrates this document. The concepts in the figure have been grouped in sets (*A to F*) to highlight the requirements addressed, as illustrated in Table 2. In the following we explain some of the most relevant concepts. Set *A* contains concepts for detecting discrepancies, checking feasibility and adapting the system. The central concept is the *Assistive Software* that performs transformations on interfaces. A *Transformation* has one source *Interface* and one target *Interface*. The *Interface* class 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). A *Controller*, e.g. a smartphone, is a device to manage *Targets*. A *Target* can be a device or service which the user wants to interact with. It sends its interface to the *Controller* via C_T_I and the interface is adapted by the controller for the user via U_C_I. The U_C_I interface is leveraged by the *User* to send actions to the controller. Such actions are translated via C_T_I, so the *Target* can execute them. The specialized interfaces allow common properties to be shared with the value partitions proposed by [19] 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* of 250%. Set *B* contains the concepts related to standards and entities for certification of AS. Set *C* includes the concepts that deal with data protection. Set *E* introduces the concept of Reviewer for the AS since a special feature of our system allows the ballot of the AS to be known, before installation. Finally, set *F* includes the concepts needed for installation, configuration and authentication of the AS, which are supplemented in part *D* with concepts to describe the properties of the SaaS (Software as a Service) since the architecture also allows to install software leveraging Web services.

Dictionary of concepts. This document describes properties and relations for each concept in the taxonomy. An excerpt, from the thirty terms described, is depicted in Table 4.

Table 4: Dictionary of concepts

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

Rules for querying the ontology. Rules make querying easier and faster, at the cost of increasing the number of assertions. Table 5 offers an example: the rule to assert that an AS can be used to ensure a correct interface between a *User* and a *Controller*. The complete catalogue contains fifteen rules for querying our ontology.

4. Design of an ASR System

The ontology described in the previous section establishes concepts defining the information needed by an ASR system. Based on this, we have conceived a sequential process, consisting of five main activities as presented in Figure 2, which addresses the requirements we established in Section 2. Such a process conforms to a high-level behavioural design of an ASR system, while the ontology

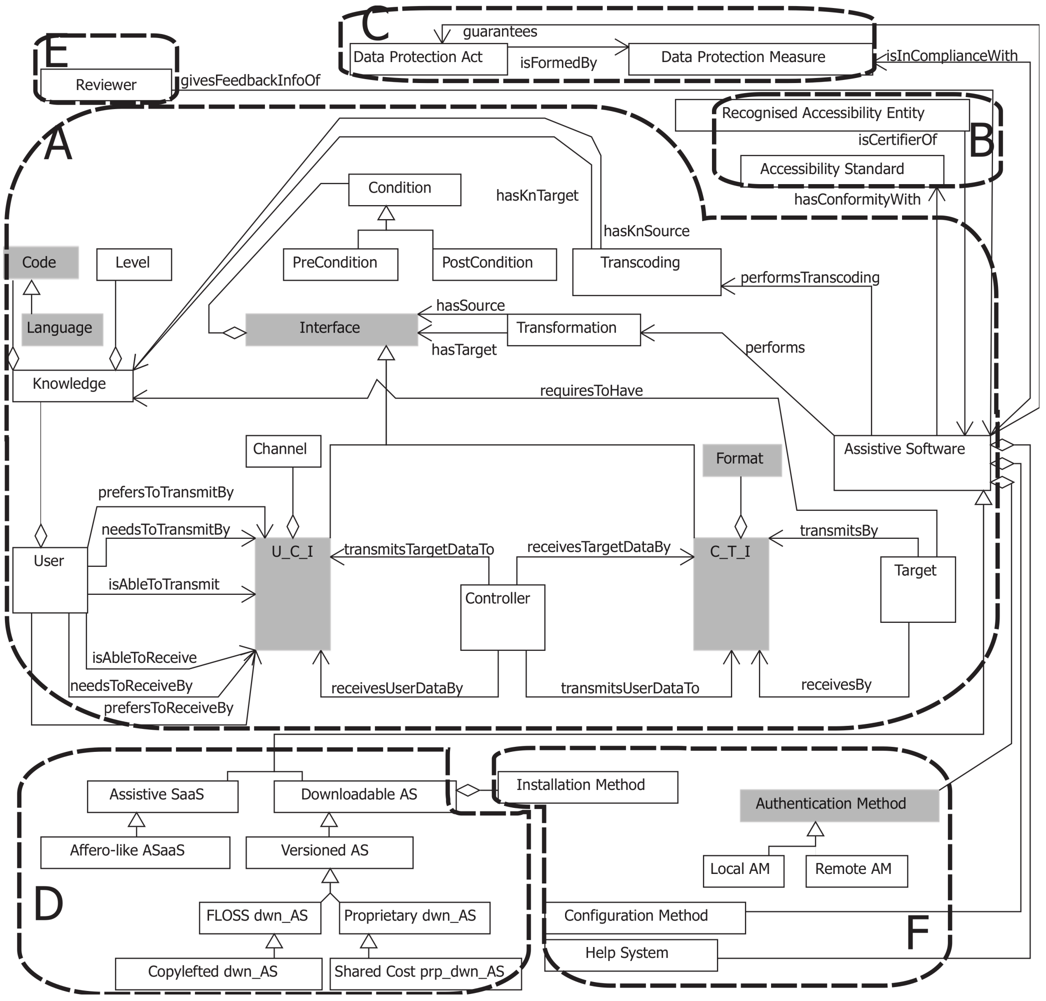


Figure 1: Taxonomy and ad-hoc binary relations diagram

Table 5: Rules for querying the ontology

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

can be seen as the structural part of the system. In the following, we describe each action contained within the process.

4.1. Detect Discrepancies

This action addresses part of requirement FR1 (*Detect "accessibility issues" for users with disability*), see Table 1. By "accessibility issues" we understand the needs, capabilities and preferences of the user for managing a controller device. "Accessibility issues" might prevent the interaction of the user with the controller device. Hence, the system needs to compare the characteristics of the interaction that the user is able to perform with those that the controller is able to emit/receive.

For detecting discrepancies, we leverage the ontology. In particular, information regarding the user's interaction capabilities is represented in the ad-hoc binary relations diagram, Figure 1-set "A". Binary relationships -*isAbleToReceive*, *isAbleToTransmit*, *needsToTransmitBy* and *needsToReceiveBy*- between *User* and U_C_I (User Controller Interface) are used to specify the user's communication capabilities for a specific channel (e.g., audio, image). The relationships *prefersToReceiveBy* and *prefersToTransmitBy* are used to specify the user's communication channel preferences. When these relationships appear empty in the ontology, i.e., there is no instance for this purpose, the system assumes that the user

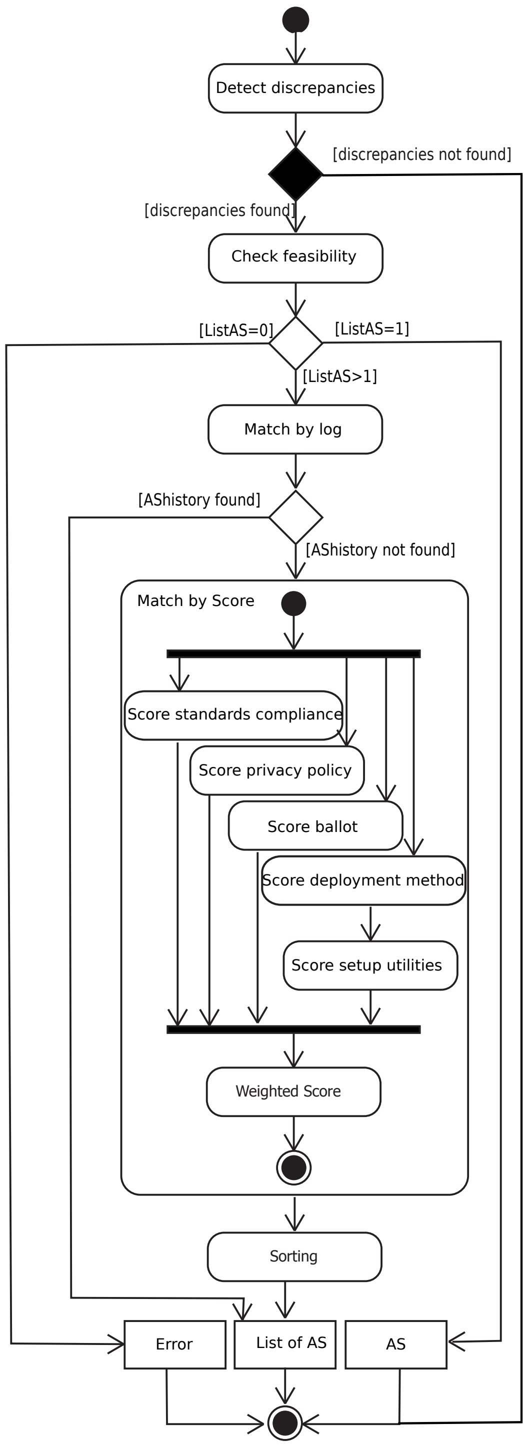


Figure 2: Process for AS recommendation

has a special need related with that channel. Hence, the system has detected a discrepancy. For example, if it has not been explicitly indicated that the user *isAbleToReceive*, e.g., audio, the system will not use this channel to communicate with the user or it will check for an AS, as explained in the next step, to cope with the audio interaction if possible.

Finally, if no discrepancies are detected for a particular channel and user then there is no need for an AS to mediate in the interaction, so the recommendation process terminates, as depicted in Figure 2.

4.2. Check Feasibility

This action completes requirement FR1 and also uses the queries of the ontology, Table 5, through the concepts of Figure 1-set "A". The goal is to analyse each discrepancy found in the previous step in order to check the feasibility of each AS to enable the interaction between the user and the controller.

Eq. 1 checks the feasibility of using an AS to emit. For interaction i, we consider the input peripheral -target device- able to support ($Interaction_i^{IP}$) and the user's capabilities and preferences for emission ($Interaction_i^{E}$). $Transformation_j$ is associated with any of the adapted interfaces present in the interaction. We annotate these as $Interaction_j^{IT}$ and $Interaction_j^{OT}$ for the input and output transformation.

$$
\begin{aligned}
&matchEmission\\
&(InputPeripheral_i, Interaction_i^E, Transformation_j)\\
&= \begin{cases}
exact & \begin{bmatrix} (Interaction_i^{IP} \equiv Interaction_j^{OT}) \wedge \\ (Interaction_j^{IT} \equiv Interaction_i^{E}) \end{bmatrix} \\
partial & \begin{bmatrix} (Interaction_j^{OT} \subseteq Interaction_i^{IP}) \wedge \\ (Interaction_i^{E} \subseteq Interaction_j^{IT}) \end{bmatrix} \\
fail & \text{otherwise}
\end{cases}
\end{aligned}
\tag{1}
$$

For the reception, we perform an equivalent matching in Eq. 2. This checks the feasibility of using an AS to receive:

$$
\begin{aligned}
&matchReception \\
&(OutputPeripheral_i, Interaction_i^R, Transformation_j) \\
&= \begin{cases}
exact & \begin{bmatrix} (Interaction_i^{OP} \equiv Interaction_i^{IT}) \wedge \\ (Interaction_i^{OT} \equiv Interaction_i^R) \end{bmatrix} \\
 & \\
partial & \begin{bmatrix} (Interaction_i^{OP} \subseteq Interaction_i^{IT}) \wedge \\ (Interaction_i^{OT} \subseteq Interaction_j^R) \end{bmatrix} \\
fail & \text{otherwise}
\end{cases}
\end{aligned}
\tag{2}
$$

In consequence, for each accessibility discrepancy we obtain three sets of AS instances, two of which might either completely solve or at least alleviate the undesired "accessibility issue" (*exact* and *partial* matches). From now on, the set of AS instances, or products, that are able to solve the accessibility problems are named Set_{AS} both in text and formulae. The following actions are intended to ascertain the most appropriate AS.

4.3. Match by Log

When the user has already employed the system to interact with the same target using the same context, then we retrieve the $AShistory$. If this set differs from Set_{AS}, then $AShistory$ is deleted, otherwise the recommendation process terminates.

4.4. Match by Score

This action is split into six and is applied to each AS in Set_{AS}. Each of the first five actions scores one important aspect of an AS: compliance with standards, privacy policy, ballot, deployment method and setup utilities. The last action is for weighting the scores.

1. *Score Standards Compliance.* The best score is obtained by those AS compliant with worldwide accessibility standards endorsed by recognized bodies. Instances of concepts in Figure 1-set "B" are queried using Eq. 3, where

$\sharp$ means cardinality:

$$\begin{aligned} score_{standard}(AS) = & \frac{\sharp(\{(x \in AccSt) \mid (x \in hCW(AS))\})}{max(\sharp(\{(x \in AccSt) \mid (x \in hCW(y))\}))} \cdot \alpha \\ & y \in Set_{AS} \\ \text{being} & \\ & AccSt = \text{Accessibility Standard} \\ & hCW(AS) = \text{is Compliant With (AS)} \end{aligned} \tag{3}$$

where α is an adjusting coefficient regarding the geographical scope of the recognized body. $\alpha = 1$ is reserved to worldwide organizations such as W3C, and values near to 0 mean local entities of minor relevance.

2. *Score Privacy Policy.* This step checks whether the AS complies with *data protection measures* endorsed by security bodies. It is important to note that, in accordance with many laws in different countries, when an *AS* complies with a *data protection act* level it also complies with certain *data protection measures*. This is taken into account here via rules to assert such measures. This is the case, for example, for the Federal Data Protection and Information Commission of Switzerland or the Data Protection Act in Spain. Instances of the concepts in Figure 1-set "C" are queried using Eq. 4:

$$\begin{aligned} score_{privacy}(AS) = & \frac{\sharp(\{(x \in DPM) \mid (x \in iCW(AS))\})}{max(\sharp(\{(x \in DPM) \mid (x \in iCW(y))\}))} \\ & y \in Set_{AS} \\ \text{being} & \\ & DPM = \text{Data Protection Measure} \\ & iCW(AS) = \text{is Compliant With (AS)} \end{aligned} \tag{4}$$

3. *Score Ballot.* This step checks how many reviews the AS has. Although there are many advantages in including reputation and trust in a recommender system [20], anonymous recommendation is also used here to safeguard the information concerning users' impairments. Eq. 5 is applied

querying instances of Figure 1-set “E”:

$$score_{ballot}(AS) = \frac{\sharp(\{(x \in R) \mid (AS \in gF(x))\})}{max(\sharp(\{(x \in R) \mid (y \in gF(x))\}))} \quad y \in Set_{AS} \tag{5}$$

being

$R =$ Reviewer

$gF(x) =$ gives Feedback Info of (x)

4. *Score Deployment Method.* This score is intentionally unbalanced because a common difficulty for people with disability is installing the AS, and this problem is to a great extent avoided by using a Software as a Service (SaaS). Eq. 6 is applied querying instances of Figure 1-set “D”.

$$score_{deployment}(AS) = \begin{cases} 1 & \text{if } AS \in \text{Assistive SaaS} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

5. *Score Setup Utilities.* This step scores the ease of access and use of the AS. All instances of the concepts in Figure 1-set “F” have predefined values, for example for Remote AM (i.e. *Authentication Method*) the values are notNecessary=1, byCookie=0.8, byPrompt=0.6, byCaptcha=0.4 and needsAssistance=0. Eq. 7 uses these values, where $v_{AS}(X)$ is the value assigned to AS for concept X.

$$score_{setup}(AS) = \begin{cases} \frac{1}{3} \cdot (v_{AS}(R\_AM) + v_{AS}(CM) + v_{AS}(HS)) & \text{if } AS \in \text{SaaS} \\ \frac{1}{4} \cdot (v_{AS}(L\_AM) + v_{AS}(CM) + v_{AS}(HS) + v_{AS}(IM)) & \text{if } AS \in \text{DW} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

being

$R\_AM =$ Remote Authentication Method

$L\_AM =$ Local Authentication Method

$CM =$ Configuration Method

$HS =$ Help System

$IM =$ Installation Method

$SaaS =$ Software as a Service

$DW =$ Downloadable Software

6. *Weighted Score.* This step weights the previous scores according to the user's preferences and the domain expert's assumptions. Eq. 8 is used:

$$WS_U(AS) = \frac{\sum_{i \in Score}(W_{Ui} \cdot W_{DEi} \cdot score_i(AS))}{\sharp\{Score\}}$$

$$\begin{aligned} &\text{being} \\ &\quad U = \text{user} \\ &\quad Score = \{standard, privacy, ballot, deployment, setup\} \\ &\quad W_{Ui} = \text{Weight that the user assigns to } i \\ &\quad W_{DEi} = \text{Weight that the domain expert assigns to } i \end{aligned} \tag{8}$$

4.5. Sort and Install

This is the final step of the whole AS recommendation process, see Figure 2. The set of AS products/services will be presented to the user according to the weighted score. The system needs to be prepared to automatically provide the first AS on the list (i.e. the most suitable AS for the given context).

5. Implementation and Deployment

We have followed the design guidelines set out in Sections 3 and 4 to develop a real and usable ASR system. Subsequently, we conducted a complete evaluation of this system, presented in Section 6. This ASR system was deployed in the INREDIS architecture, which was especially designed for adapting interfaces to people with special needs. The remainder of this section describes the deployment of the ASR system and highlights some important implementation issues.

5.1. Deployment in the INREDIS Architecture

The INREDIS project[7] (INterfaces for RElations between Environment and people with DISabilities) aimed to develop interaction channels between people with some kind of special need and their context, where the targets were auto-discoverable devices and services. Figure 3 presents a simplified view of the whole INREDIS architecture. More than 200 researchers from 14 Spanish companies and 19 research organizations collaborated to carry out the INREDIS project during 48 months with a budget of €23.6 millions. The ASR system consumed an

[7]http://www.inredis.es/default.aspx

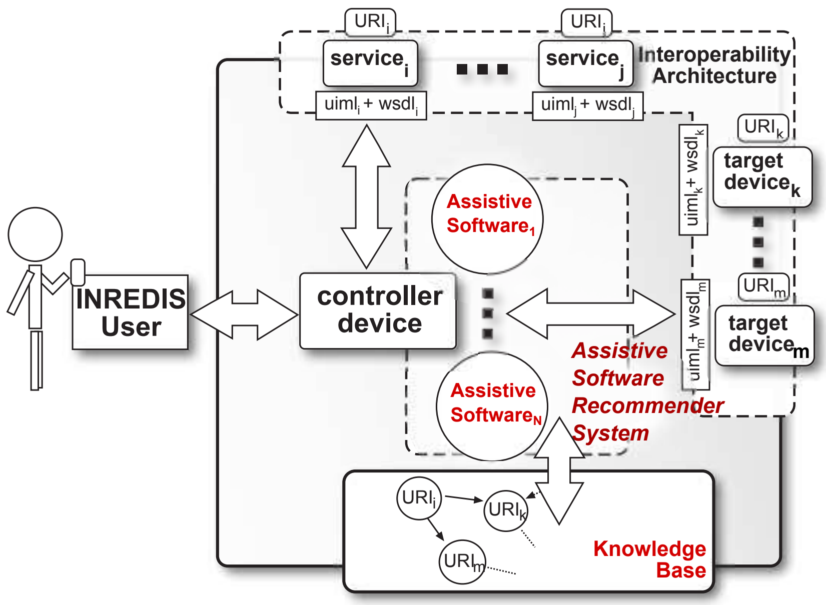
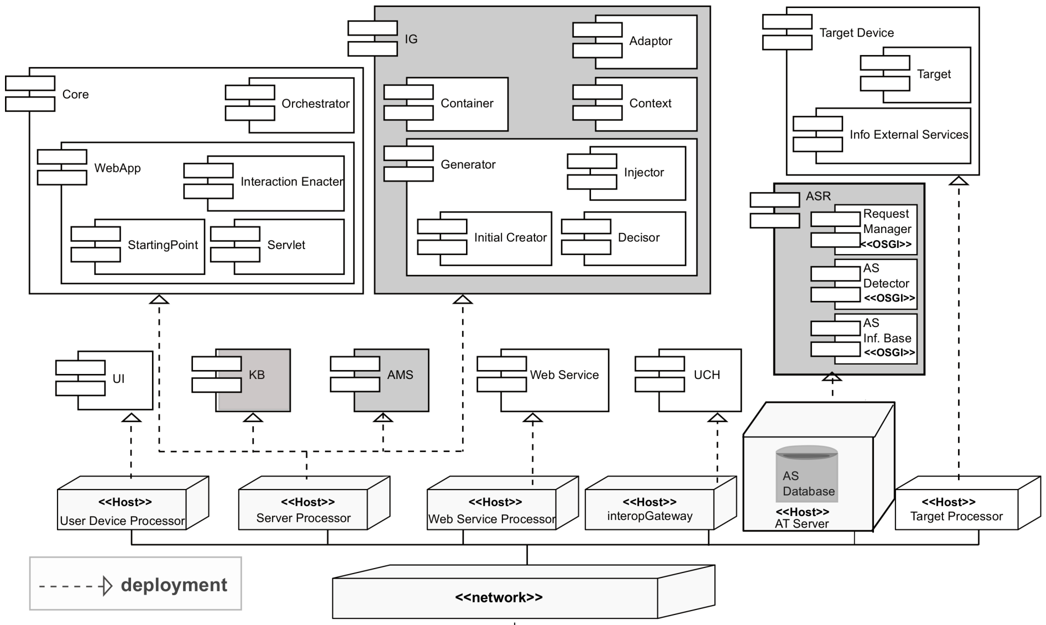


Figure 3: INREDIS architecture

important part of the INREDIS resources. It was conceived as a universal solution capable of providing people with disabilities and elderly people with accessible and personalized interfaces according to their preferences and needs.

Let us return to the scenario introduced in Section 2 to clarify how the ASR system fits into the INREDIS architecture. *Tom wants to operate the air conditioning system (AC) in his room. Firstly, he accesses the INREDIS architecture with his nickname and password using his smartphone (controller device). Secondly, the ASR system helps to select and install interfaces, according to his disabilities, for controlling those new services and devices not already managed by his smartphone. A screen in his smartphone displays the available target devices and services, grouped by environment, e.g. "Smart Home", "Products and Services" and "Health Care", among others. These devices and services depend on the user's location, in this case the hotel. Obviously, all screen dialogs need to be accessible and adapted to the specific needs and preferences of the user. Tom navigates through the screen dialogs until he identifies the AC, for instance, in the "Smart Home".*

The INREDIS architecture is an event-driven and service-oriented architecture. As outlined in Figure 3, it considers a user surrounded by a set of devices and services, the architecture uses different interoperability protocols and frame-



AMS	Adaptive Modeling Server	KB	Knowledge Base
AS	Assistive Software	OSGI	Open Services Gateway Initiative
ASR	AS Recommender	UI	User Interface
AT Server	Assistive Technologies Server	UCH	Universal Control Hub
IG	Interface Generator	WebApp	Web Application

Figure 4: INREDIS deployment

works (such as URC [21], OSGi or Web Services standards) to facilitate the interaction in accordance with the specific nature of each target device/service. When there are "accessibility issues" between the user and the controller device, then the ASR system starts executing various actions: it detects discrepancies, it discovers AS instances automatically and it launches the rest of the processes described in Section 4.

The most relevant software and hardware components for the INREDIS architecture are depicted in the deployment diagram in Figure 4. Those software components that implement the ASR system are highlighted in grey, specifically the ASR component itself, the Knowledge Base (KB in Fig. 4), the Adaptive Modelling Server (AMS in Fig. 4) and the Interface Generator (IG in Fig. 4). The following subsection describes interesting implementation issues for each component.

5.2. Implementation within INREDIS

ASR Component. This component implements the processes described in Section 4. Hence, it provides automatic discovery and configuration of assistive technologies, in a smart and transparent fashion reducing the existing accessibility gap that may exist between the users and their universal controller device. This component aggregates three OSGi [22] components/bundles and a database component deployed in the same server, as follows.

Request Manager. Being an OSGi bundle this component shows a Web service that is invoked by INREDIS when an AS product is needed. INREDIS provides several parameters (user identifier, user device and target service or device) and obtains a list, of the available AS products, that matches the request. This component manages the concurrency of the requests and sends them to the AS Detector. Once it gets an answer, it then creates the list for INREDIS that also includes the configuration of each AS product for the current user.

AS Detector. This manages the requests sent by the Request Manager and queries the Knowledge Base (KB) for the most suitable AS products. Once it gets a response, it then queries the AS Information Base to obtain the configuration for the current user for each AS retrieved. All this information is delivered to the Request Manager. The sequence diagram in Figure 5 details the messages exchanged among all the components involved in this interaction.

AS Information Base. This component is an information repository for the URLs of the AS products. Moreover, it stores information about each AS product's configuration (e.g., login scripts, configuration scripts). Irrespective of whether it is used within INREDIS or standalone, it always provides the same interfaces and behaviour. It also requests the AS Database for the specific AS configuration for a given user.

AS Database. This stores for each user the configuration s/he has selected for each AS already used. Given that the underlying data model of the information stored here is not very complex, INREDIS decided to implement this database with MySQL [23].

Knowledge Base (KB). The Knowledge Base implements the ontology described in Section 3. Consequently, it stores all the ontologies that collect formal descriptions of the elements in the INREDIS domain (e.g., users, AS or devices) and their instances. It uses descriptive logic and provides mechanisms for reasoning

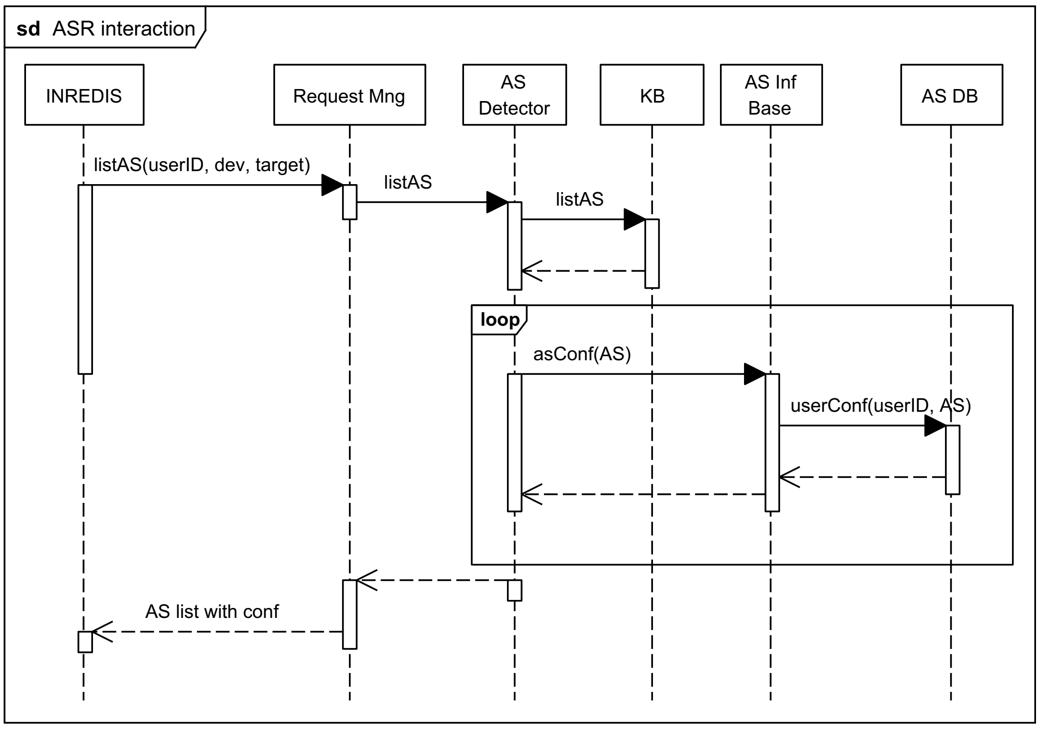


Figure 5: ASR interaction

and querying, enabling the intelligent behaviour of the architecture, as described in Section 3. It was implemented using OWL [24], SPARQL [25], Drools [26] and Pellet [27].

Adaptive Modelling Server (AMS). This keeps the KB content updated using information from different heterogeneous sources (application context, user interaction logs or complex events processing).

Interface Generator (IG). This component adapts interfaces expressed in a generic and abstract language, a subset of the User Interface Markup Language [28] (UIML), into concrete utilizable and accessible language (implemented in XHTML [29]). Transformations are carried out using XSLT [30] which allows different formats (e.g., XML, pdf, plain text) to be created from XML files. The IG cooperates with:

1. The KB, which provides support for storing the knowledge needed by the IG to match the user needs with the interface to be generated. In particular,

the KB provides methods, rules and reasoning mechanisms for establishing complex relationships between the concepts stored. Moreover, the KB stores the XSLT transformation for each AS.

2. The ASR component for encapsulating the functionality of the AS finally selected - REST and SOAP technologies are used to publish the AS as a Web service. For example, consider that the selected AS product transforms text-to-speech for blind people. The IG prepares both the text file and a call to the REST service with this text (including, for example, the language of the user and the speed of conversion). Then, the ASR receives a URI with a WAV file inserted in the resulting interface.

Management of the needs of the user. Some of the components presented in this subsection cooperate to manage the needs and preferences of the user in the AS selection. Firstly, a prototype was developed for self-detecting user's needs and capabilities. This prototype follows the model of accessibility proposed by the ISO 24756:2009 [9] standard. This standard defines a framework for specifying a common access profile (CAP) of needs and capabilities of users, including access supported by assistive technologies. Moreover, the standard specifies methods to apply a CAP. Therefore, each INREDIS user initially needs to execute this prototype, and consequently the system stores the CAP generated by the prototype in the KB. When a user logs onto INREDIS, then his/her CAP is retrieved and the interfaces of his/her controller are adapted accordingly. For example, if the user has pathological or degenerative myopia, the interface could offer an iZoom magnifier. Additionally, the system allows the user to change this magnifier for another one of his/her choice. Such interactions are continuously managed by the AMS to update the CAP. Furthermore, the CAP is also used by the ASR to query the KB when the ASR needs to select an AS.

6. Evaluation

The implementation of the ASR system and its deployment in INREDIS provided us with a framework in which to evaluate the fulfilment of the requirements set out in Section 2. The evaluation of the functional requirements, described in Subsection 6.1, assesses whether the design we propose for ASR systems adequately meets the needs of users for obtaining and installing the desired AS automatically. On the other hand, the evaluation of the non functional requirements discussed in Subsection 6.2, assesses the fulfillment of the Nielsen principles and the scalability of the specific implementation carried out.

In [31] the complete INREDIS architecture was experimentally tested through a set of user controlled tests. The tests considered diverse user preferences and disability profiles (cognitive disabilities, deafness, partial blindness, congenital blindness, etc.). The difficulties of carrying out real experiments with elderly people and people with special needs quickly became apparent and severely limited the testing. The number of users never exceeded twelve due to logistical difficulties, as described in [11]. For example, some users needed caregivers or special assistance to attend to their disabilities. In most cases, for twelve users a team of a least twenty four people was required to work in a small smart-home specially rented for the tests. Moreover, the relevant Spanish legal regulations governing such tests obviously had to be complied with.

6.1. Functional Requirements Evaluation

The inconveniences of real experimentation prevented us from testing each of the six functional requirements separately. Instead, we decided to test whether the ASR system was able to obtain the same recommendation as would be obtained by a domain expert following EASTIN guidelines [32], i.e., whether the AS product installed would be the same as that advised by a domain expert. Note that by achieving this objective, we ensure that the system detects “accessibility issues” correctly (FR1), provides a list of AS products where the relevant ones are presented (FR3), correctly advises the user (FR4) and installs the software (FR6). The other two functional requirements were also successfully tested during the evaluation.

We selected 8 (real) impaired users with different impairment profiles (2 visual, 2 multifunctional-elderly, 2 motor and 2 hearing impaired people). In the tests we included features such as Google translator[8], Text-To-Speech (TTS), specifically Live555 media server[9] and e-speak[10], or a sign language avatar developed in [33]. Videos of some of these tests are available online [34]. We populated the Knowledge Base (KB) with 41 AS products from EASTIN covering all the impairments to be tested (i.e., visual, multifunctional, motor and auditory). On this basis, the AS product selected by the users with the ASR system always matched the selection by the expert using the EASTIN search engine.

Additionally, the ASR system was evaluated with the same 8 users, but including in the KB some AS products deployed as software as a service (SaaS), which

[8]`http://translate.google.com/about/intl/en_ALL/`

[9]`http://www.live555.com/mediaServer/`

[10]`http://espeak.sourceforge.net/`

were not available in the EASTIN catalogues. In this case, the recommendations of our ASR system were consistent with the recommendations given by domain experts. Indeed, we assume that with a fully populated KB, the self-learning capability of the ASR system would make better recommendations than a domain expert, since it is able to compare more AS products taking into account more features.

Finally, we leveraged this experimentation step in order to set values to the variables W_{Ui} and W_{DEi}, that were used in Equation 8. These represent the weight assigned by the users and the domain expert to each feature (standards, privacy, ballot, etc.) in the selection process. Our hypothesis was that by using a small AS dataset and focusing on a specific impairment, we would obtain better values than using a bigger AS dataset supporting different impairments. We described 4 users' profiles with different visual-impairments (colour blindness, achromatopsia, tritanopia and totally blind). We specified 4 different contexts for each user, such as different languages, then provided 16 (i.e., 4×4) inputs to the ASR system. We also populated the KB with 20 AS products for visual limitation which appear in the EASTIN guidelines. In all the experiments the weight assigned by the user, W_{Ui}, was coincident -the differences were extremely small - with the weight assigned by the expert, W_{DEi}, which means that the ASR selection should match users' preferences and needs.

6.2. Non Functional Evaluation

Requirement NFR1 established the need for an ASR system to be flexible, easy to use and communicative according to Nielsen principles. A complete study of user experience and user satisfaction in the context of INREDIS can be found in the works [11] and [35]. These works concluded that users perceived the facilities as comfortable and adequate. Regarding the ASR system, the technology used was accepted as a convenience by some participants and as necessary products and services by others, but the usability was assessed as adequate.

Regarding requirements NFR2 and NFR3, [36] reports a complete study about the performance (i.e., response time and scalability) of the INREDIS architecture. In the following, we summarize the most interesting results reported in that work for the ASR system. However, we strongly recommend reading the report in [36] to fully understand how the non functional evaluation was carried out.

The difficulties of the real experimentation described at the beginning of this section were aggravated during this evaluation step. In this case, in order to obtain results about the performance of our system, we had to instrument the code and manage the intricacies of the INREDIS architecture while simultaneously testing with users. As a consequence, the number of concurrent users with disabilities could never exceed 5 for the performance evaluation in the smart home where the tests were carried out. As reported in [36], we decided to overcome this problem by using models of the system. Models can represent the system in a variety of hypothetical situations and can perform evaluation at a lower cost. In particular, we followed a mature discipline known as Software Performance Engineering [37] and we relied on stochastic models, specifically Generalized Stochastic Petri Nets (GSPN) [38]. As explained in [36], we obtained the GSPN models directly from the ASR system design, and our first objective was the validation of these performance models. Later, we obtained results from the valid models to assess an optimal system configuration to meet the response time requirement. We recall that following Nielsen [10], we decided that periods of around ten seconds could be considered as acceptable response times.

Figure 6 depicts the measured average response times obtained in the user testing phase for 5 users in the smart home. We then reproduced these experiments using the performance models and we obtained the results shown in Figure 7. We observed that the differences between our models and the real experimentation with up to 5 users were around five percent and that the tendencies in the graphs were similar. We therefore assumed that our performance models would be useful for experiments not initially feasible to carry out within the real setting. Note that using models we obtained results for one hundred users, which was sufficient for our purposes. Indeed, we could have obtained results for larger populations using the same GSPN models but increasing the workload. We can observe in Figure 7 that the response time of the system is below ten seconds until it reaches forty concurrent users. For more than forty users the ASR system does not meet the response time requirement.

In [36] we developed several alternatives for improving responsiveness including resource replication, performance patterns and performance antipatterns. The most significant improvement was achieved when we detected The Ramp antipattern in the ASR system. Antipatterns extend the notion of patterns to capture design errors and their solution [39]. The problem arose when we populated the Knowledge Base (KB) with ten thousand AS products, since the ASR incrementally searches the KB. To solve this antipattern, Smith and Williams [40] propose selecting another search algorithm, more appropriate for large amounts of data. The original searching algorithm was based on a simple filtered search in SPARQL. This search was improved by designing a specific “recommend” operator, as Levandoski et al [41] suggest. Thus, the response time for a search improved by 33%, independently of the number of users. Finally, our GSPN models

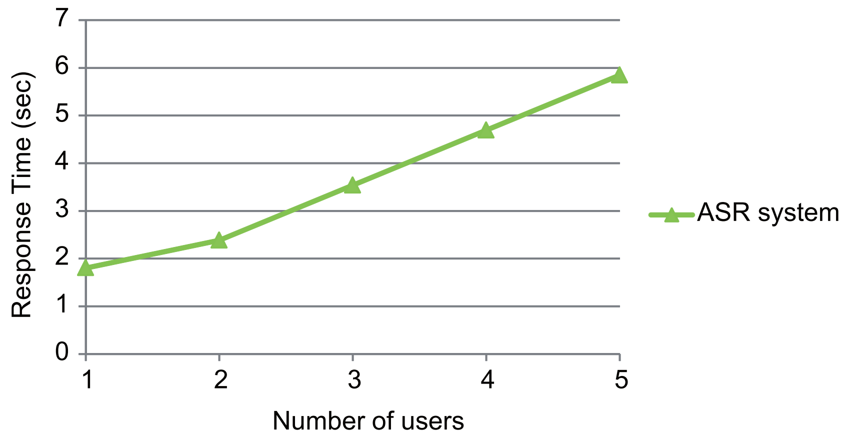


Figure 6: Response time of the ASR system -empirical results-

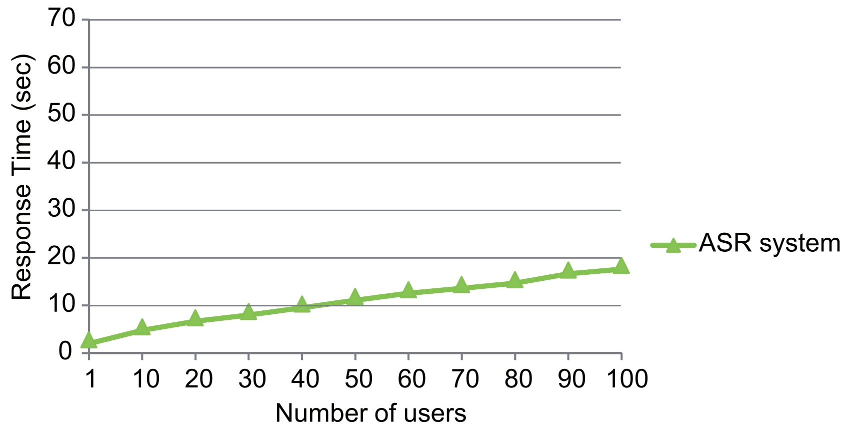


Figure 7: Response time of the ASR system -using models-

obtained an optimal system configuration for the INREDIS architecture that fitted the ASR. Figure 8 depicts these results. We then applied these improvements to the real architecture implementation and repeated the experiments in the smart home with 5 users. The results are shown in Figure 9. They match those in Figure 8 - from 1 user up to 5 users. Regarding the scalability advocated by NFR3, the results in Figure 8 indicate that the system scales well for 100 users, which is an acceptable number for a smart home or even for larger facilities.

7. Related Work

Given that we have not found any works specifically reporting on how to de-

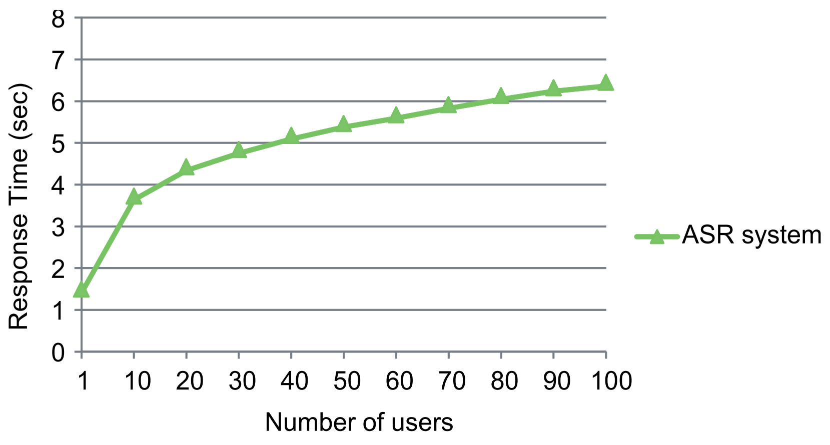


Figure 8: Response time and scalability of the ASR system -using models-

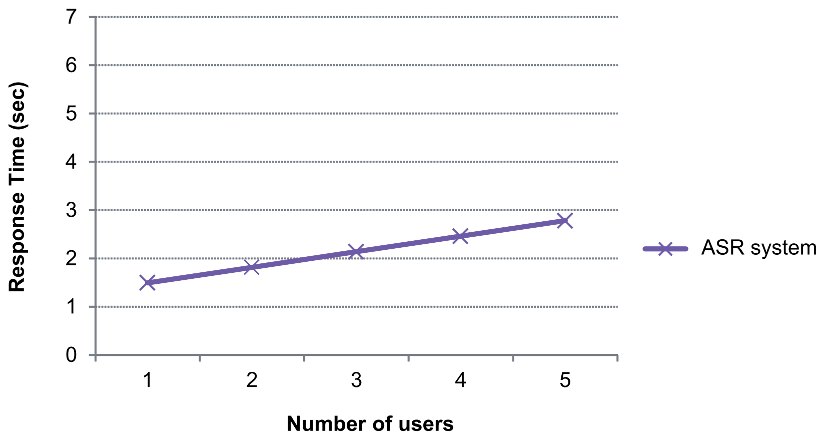


Figure 9: Response time of the optimized ASR system -empirical results-

velop automated systems which can perform the task of selecting assistive software (AS) products, we review those related works which have a bearing on AS cataloguing, selection or recommendation.

The ISO 9999 [42] standard establishes a classification of assistive products, AS included. EASTIN [32] is a European initiative that uses ISO 9999 to develop its databases in order to compile a list of assistive products (not only AS). EASTIN also publishes textual guidelines, written by domain experts, that guide users in order to reduce the number of AS products and services that they have to try. Other organizations, such as ONCE in Spain, make their own AS recommendations.

In [43] an ontology is proposed to establish a list of accessibility requirement specifications. In [44] a repository of ontologies is proposed aimed at raising metadata interoperability across assistive technology repositories. While the former does not consider the user's capabilities and lacks AS specification and categorization, the latter does not consider the user's needs and context-aware aspects. In [45] an ontology-centered design is presented to model the environment in smart homes, but issues such as user capabilities and AS-user interactions are not considered.

The work of Kaklanis [46, 47] presents a Semantic Framework for Content and Solutions (SEMA) that has been developed within the Cloud4all FP7 EC project. It focuses on user interface adaptation as it enhances inference capabilities to match user needs with the corresponding configurations of different assistive technologies using rules. This work provides a high-level modeling of content-related information of ICT solutions, platforms and devices. A framework was developed that enables the semantic representation of assistive technologies, and the Semantic Alignment Tool aimed at providing a common interface to all interested stakeholders that was designed to include all the applications/solutions in the Cloud4all infrastructure.

The work in [48] develops an ontology, using the Delphi method, for one class of assistive technology, namely physically controllable pointing devices. Based on this ontology, OSCAR [49] (Ontology Supported Computerized Assistive Technology Recommender) is a proof-of-concept case of CDSS (Clinical Decision Support System) designed to help clinicians in the decision making process for selecting appropriate physically controllable pointing devices. OSCAR presents a novel, knowledge-driven approach that uses communication technologies, data, documents, knowledge and models to identify problems and solve them. It consists of a systematic set of coded algorithms designed to use a computerized clinical knowledge base to propose matches between client characteristics, needs and abilities and appropriate solutions. Recommendations are then presented to the clinician.

Finally, Kadouche et al. [50] propose a semantic framework to improve environment services for people with special needs. However, there are two main differences: firstly, in our approach we do the reasoning at a class level, which makes our approach more expressive and scalable; secondly, we take into account the AS recommender process whereas the framework described in [50] does not.

8. Conclusion and Future Work

Although there are works in the literature reporting the development of recommender systems, the design of systems for selecting AS has not yet been addressed. However, we believe that this topic is of increasing importance since, in the near future, smart environments for people with special needs will become established in large facilities such as nursing homes, hospitals or leisure areas. In this regard, the main contribution of our work is an AS recommender system which enables existing interoperability architectures, such as INREDIS [8] or MobiEureka [51], to automatically select the most suitable AS for a given interaction with a specific electronic target device taking into account the user's context (user, controller device and target device) and considering the disability in question.

This work focuses on practical experience for practitioners in the field of assistive technologies, particulary in relation to software requirements, design and implementation. We believe that this work can guide practitioners in the design of an ontology for ASR systems and in the design of the processes involved. The work also addresses software integration and system issues, which can help practitioners in questions of deployment. We have thoroughly validated the proposal by implementing the system following the design presented and integrating the system in a real architecture, INREDIS. Last but not least, the proposal has also been validated by assessment; in fact both the functional and the non-functional requirements have been assessed.

The performance evaluation has ensured that no extra overhead will be involved as a result of bad-design practices, and that the system scales at least as well as the semantic query engine. The performance models enable the system to be tested in hypothetical situations, which could be difficult or expensive to carry out in real situations. For example, we could forecast the behaviour of the system when used by 100 users, which would otherwise involve an expensive non-viable experiment.

For future work we have identified a further promising line of research. We aim to use machine learning techniques to provide: a) user profile updates according to user AS recommendations and b) the incorporation of sensors as an input to recommend the best AS under noisy conditions (e.g., a user with restricted vision may be able to read from the mobile screen, except outdoors on a very sunny day). Advances in mobile technologies allow make the collection of user context information feasible, and this issue also needs to be considered.

Acknowledgements

The research described in this paper arises from the Spanish research project INREDIS (INterfaces for RElations between Environment and people with DISabilities). The authors wish to thank the researchers who have been periodically reviewing the AS ontology specification and the recommender system. INREDIS is led by Technosite and funded by the CDTI (Industrial Technology Development Centre) under the CENIT (National Strategic Technical Research Consortia) Programme, within the framework of the Spanish governments INGENIO 2010 initiative. The opinions expressed in this paper are those of the authors and are not necessarily those of the INREDIS project partners or of the CDTI.

The authors want to thank the anonymous reviewers for their valuable comments that have largely improved the paper.

José Merseguer has been supported by the European Commission grant agreement no. H2020-644869 (DICE).

References

[1] C. Stephanidis, Adaptive Techniques for Universal Access, User Modeling and User-Adapted Interaction **11** (1-2) (2001) 159–179.

[2] World Health Organization, ICF: International classification of functioning, disability and health, World Health Organization, 2001.

[3] G. Zimmermann, G. Vanderheiden, Accessible design and testing in the application development process: considerations for an integrated approach, Universal Access in the Information Society 7 (1) (2008) 117–128.

[4] G. Margetis, M. Antona, S. Ntoa, C. Stephanidis, Towards Accessibility in Ambient Intelligence Environments, Ambient Intelligence (2012) 328–337.

[5] R. Andrich, A. Caracciolo, I. Johnson, Individual assessment for assistive technology solutions: Reflections on a thirty-year experience, Technology and Disability 25 (3) (2013) 147–158. doi:10.3233/TAD-130379.

[6] R. Andrich, N.-E. Mathiassen, E.-J. Hoogerwerf, G. J. Gelderblom, Service delivery systems for assistive technology in Europe: An AAATE/EASTIN position paper, Technology and Disability 25 (3) (2013) 127–146. doi:10.3233/TAD-130381.

[7] O. Nee, A. Hein, T. Gorath, N. Hulsmann, G. Laleci, M. Yuksel, M. Olduz, I. Tasyurt, U. Orhan, A. Dogac, A. Fruntelata, S. Ghiorghe, R. Ludwig, Saphire: intelligent healthcare monitoring based on semantic interoperability platform: pilot applications, Communications, IET 2 (2) (2008) 192–201. doi:10.1049/iet-com:20060699.

[8] INREDIS, INterfaces for RElations between Environment and people with DISabilities, `http://www.inredis.es/` (2011).

[9] ISO/IEC, 24756:2009 - Framework for specifying a common access profile (CAP) of needs and capabilities of users, systems, and their environments, ISO/IEC 24756, International Standard Organization (2009).

[10] J. Nielsen, Usability Engineering, Morgan Kaufmann Publishers Inc., 1993.

[11] F. Sainz, J. Casacuberta, M. Díaz, J. Madrid, Evaluation of an Accessible Home Control and Telecare System, in: Proc. 13rd Human-Computer Interaction (INTERACT'11), Vol. 6949 of LNCS, Springer-Verlag, 2011, pp. 527–530.

[12] R. Giménez, M. Pous, F. Rico-Novella, Securing an Interoperability Architecture for Home and Urban Networking: Implementation of the Security Aspects in the INREDIS Interoperability Architecture, in: Proc. 26th Int. Conf. on Advanced Information Networking and Applications Workshops (WAINA'12), Vol. 0, IEEE Computer Society, 2012, pp. 714–719.

[13] A. Murua, I. González, E. Gómez-Martínez, Cloud-based Assistive Technology Services, in: Proc. Federated Conf. on Computer Science and Information Systems (FedCSIS'11), 2011, pp. 985–989.

[14] M. Pous, C. Serra-Vallmitjana, R. Giménez, M. Torrent-Moreno, D. Boix, Enhancing accessibility: Mobile to ATM case study, in: Proc. IEEE Consumer Communications and Networking Conf. (CCNC'12), IEEE Computer Society, 2012, pp. 404–408.

[15] R. Studer, V. Benjamins, D. Fensel, Knowledge Engineering: Principles and Methods, Data Knowl. Eng. **25** (1-2) (1998) 161–197.

[16] K. Masuwa-Morgan, P. Burrell, Justification of the need for an ontology for accessibility requirements (Theoretic framework), Interacting with Computers **16** (3) (2004) 523–555.

[17] Ó. Corcho, M. Fernández-López, A. Gómez-Pérez, Methodologies, tools and languages for building ontologies: Where is their meeting point?, Data Knowl. Eng. **46** (1) (2003) 41–64.

[18] World Wide Web Consortium (W3C), OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/ (2004).

[19] M. Horridge, H. Knublauch, A. Rector, R. Stevens, C. Wroe, A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools, University Of Manchester, 1st Edition, http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/ (2004).

[20] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, Decision Support Systems 43 (2) (2007) 618 – 644, emerging Issues in Collaborative Commerce.

[21] G. Zimmermann, G. C. Vanderheiden, The Universal Control Hub: An Open Platform for Remote User Interfaces in the Digital Home, in: Proc. of Int. Conf. Human-Computer Interaction, Vol. 4551 of Lecture Notes in Computer Science, Springer, 2007, pp. 1040–1049.

[22] OSGi Alliance, Open Services Gateway Initiative, http://www.osgi.org/Main/HomePage (March 1999).

[23] Oracle Corporation, MySQL, http://www.mysql.com/ (2014).

[24] W3C, OWL 2 Web Ontology Language, http://www.w3.org/TR/owl2-overview/ (December 2012).

[25] E. Prud'hommeaux, A. Seaborne, SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/ (2006).

[26] RedHat, Drools Business Logic integration Platform, http://drools.jboss.org/ (2014).

[27] Clark and Parsia, Pellet: OWL 2 Reasoner for Java, http://clarkparsia.com/pellet/ (2014).

[28] C. Phanouriou, UIML: A Device-Independent User Interface Markup Language, Tech. rep., Virginia Polytechnic Institute and State University (2000).

[29] eXtensible HyperText Markup Language, `http://www.xhtml.org/` (2010).

[30] W3C, XSLT (Extensible Stylesheet Language Transformations), `http://www.w3.org/TR/xslt/` (November 1999).

[31] E. Catalán, M. Catalán, Performance Evaluation of the INREDIS framework, Technical report, Departament d'Enginyeria Telemàtica, Universitat Politècnica de Catalunya, Internal report not-available outside the institution. (2010).

[32] V. Gower, R. Andrich, A. Agnoletto, P. Winkelmann, T. Lyhne, R. Rozis, G. Thurmair, The European Assistive Technology Information Portal (EASTIN): Improving Usability through Language Technologies, in: Proc. of 13th Int. Conf Computers Helping People with Special Needs (ICCHP 2012), Springer, 2012, pp. 215–222.

[33] J. Porta, F. J. López-Colino, J. Tejedor, J. Colás, A rule-based translation from written Spanish to Spanish Sign Language glosses, Computer Speech & Language 28 (3) (2014) 788–811.

[34] INREDIS, Video Page, `http://www.youtube.com/user/INREDIS/` (2011).

[35] J. Casacuberta, F. Sainz, J. Madrid, Evaluation of an Inclusive Smart Home Technology System, in: Ambient Assisted Living and Home Care, Vol. 7657 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 316–319.

[36] E. Gómez-Martínez, R. González-Cabero, J. Merseguer, Performance assessment of an architecture with adaptative interfaces for people with special needs, Empirical Software Engineering 19 (6) (2014) 1967–2018, `http://dx.doi.org/10.1007/s10664-013-9297-1`.

[37] C. Smith, Performance Engineering of Software Systems, 1st Edition, The Sei Series in Software Engineering, Addison–Wesley, 1990.

[38] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, Modelling with Generalized Stochastic Petri Nets, John Wiley Series in Parallel Computing, 1995.

[39] W. Brown, R. Malveau, H. McCormick, T. Mowbray, AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis, John Wiley, 1998.

[40] C. U. Smith, L. G. Williams, New software performance antipatterns: More ways to shoot yourself in the foot, in: Proc. 28th Int. Conf. Computer Measurement Group (CMG'02), 2002, pp. 667–674.

[41] J. J. Levandoski, M. D. Ekstrand, M. Ludwig, A. Eldawy, M. F. Mokbel, J. Riedl, RecBench: Benchmarks for Evaluating Performance of Recommender System Architectures, PVLDB 4 (11) (2011) 911–920.

[42] ISO, 9999:2011 - Assistive products for persons with disability – Classification and terminology, ISO 9999, International Standard Organization (2011).

[43] K. Masuwa-Morgan, Introducing AccessOnto: Ontology for Accessibility Requirements Specification, in: Int. Workshop on Ontologies in Interactive Systems, Vol. 00, IEEE Computer Society, 2008, pp. 33–38.

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

[45] M. Klein, A. Schmidt, R. Lauer, Ontology-Centred Design of an Ambient Middleware for Assisted Living: The Case of SOPRANO, Towards Ambient Intelligence Methods for Cooperating Ensembles in Ubiquitous Environments **10**.

[46] N. Kaklanis, K. Votis, K. Giannoutakis, D. Tzovaras, A semantic framework for assistive technologies description to strengthen UI adaptation, in: C. Stephanidis, M. Antona (Eds.), Universal Access in Human-Computer Interaction. Design and Development Methods for Universal Access, Vol. 8513 of Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 236–245.

[47] N. Kaklanis, K. Votis, K. Giannoutakis, D. Tzovaras, V. Gower, R. Andrich, A unified semantic framework for detailed description of assistive technologies based on the EASTIN taxonomy, in: K. Miesenberger, D. Fels, D. Archambault, P. Pez, W. Zagler (Eds.), Computers Helping People with Special Needs, Vol. 8548 of Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 275–282.

[48] A. Danial-Saad, T. Kuflik, P. L. Weiss, N. Schreuer, Building an ontology for assistive technology using the Delphi method, Disability and Rehabilitation: Assistive Technology 8 (4) (2013) 275–286.

[49] A. Danial-Saad, T. Kuflikb, P. Weiss, Oscar: Clinical decision support system for prescription of assistive technology to people with disabilities, in: P. Encarnao, L. Azevedo, G. Gelderblom, N. A, N. Mathiassen (Eds.), Assistive Technology: From Research to Practice, IOS press, 2013, pp. 241–246.

[50] R. Kadouche, B. Abdulrazak, S. Giroux, M. Mokhtari, Disability centered approach in smart space management, Int. Journal of Smart Home (IJSH) **3** (3) (2009) 13–26.

[51] D. Vergados, Service personalization for assistive living in a mobile ambient healthcare-networked environment, Personal Ubiquitous Comput. **14** (2010) 575–590.