MyUI: Mainstreaming Accessibility through Synergistic User Modelling and Adaptability

FP7-ICT-2009-4-248606

Context Ontology, User Modelling Concept and Context Management Architecture

Public Document / VUMS Cluster Document

Deliverable number	D1.1	**Date of delivery**	31-01-2011
Status	Final	**Type**	Report
Workpackage	WP1- User and Context Modelling		
Authors	FZI (Peter Wolf, Oliver Strnad, Heiko Haller, Andreas Schmidt), FHG-IAO (Matthias Peißner), UC3M (José Alberto Hernández), PCL (Roy van de Korput)		
Keywords	Context; Context Management; Architecture; Ontology; User Model, User Profile		
Abstract	This deliverable describes the MyUI Context Ontology and the Context Management Architecture. A major part of the document is dealing with the dynamic MyUI User Profile which serves as the basis for MyUI's adaptive user interfaces.		

Table of Contents

1. Introduction

MyUI aims at providing support for user interfaces that dynamically adapt to the needs of the user and the characteristics of the user's environment, i.e., the **context** of a user. Towards that end, we need the following

- A structured **representation** of the context, often labelled as "user model" and "environment model". For this representation, we need to define the formalism, available features and possible values. Within MyUI we have chosen a ontology-based approach allowing the project team to collaboratively develop the **context ontology**.
- Methods and algorithms for deriving the context (at an abstraction level that forms the basis for adaptation strategy) from low-level input (physical sensors, user actions). This process has often been called "user modelling", context "abstraction", "aggregation", or "augmentation" (see [SchmidtContext]). In this deliverable, we call this context augmentation.
- A technical infrastructure for providing other components in the MyUI system with a consistent view on the current context of the user using the vocabulary and constraints of the context ontology. This is achieved through the **context management implementation**.

1.1 Responding to review recommendation

Following the M6 review results and recommendations, the overall plan of WP1 has shifted from a more analytic approach where algorithms are designed based on analysis of existing knowledge and user studies towards a rapid-prototyping approach in which we develop a running system as a basis for experimenting and gathering runtime experiences. Therefore, we have concentrated our work in year 1 on (a) the fundamentals of modelling as the basic enabler and (b) on providing components for the first demonstration for M12. To take advantage of this approach, the development of methods for deriving context information from low-level sensor information has been limited to basic heuristics in year 1 and will be more experimental within year 2.

Furthermore, the approach to ontology modelling has been thoroughly reassessed, resulting in a more lightweight and less diagnosis-centred approach that has been found to be more suitable for the problem at hand.

1.2 General Approach to MyUI's Context Management Solution and Outline of this document

As described in the previous section, this deliverable aims for 3 distinct results: description of the general method and ideas behind the MyUI context and user profile management framework, presentation and discussion of the current MyUI context modelling with a focus und the user modelling and a brief presentation of the implementation of the context management solution.

As shown in chapter 3.1, work on this task started with the analysis of the requirements identified in deliverable D2, as well as the scenarios described in D4.1. This resulted in the selection of the email client application scenario as first demonstrator, since this scenario is able to exemplify all major contributions of the project without requiring extensive implementation efforts. Based on technical background knowledge in the context management and user modelling domain from projects like LIP, Agent-Dysl, SOPRANO, and openAAL, a first approach to user profiling based on ontology based context management was derived for this scenario and discussed within the project consortium. This was the starting point to a process of iterative refinement of the technical concepts, where each iteration result was presented to the consortium and where feedback was gathered and incorporated into the concept; until a stable technical concept was reached.
This stable version of the technical concept is presented in the upcoming sections of this deliverable. While section 2 gives an overview and discusses the general concepts underlying the

MyUI context management framework, section 3 gives more details towards the context ontology and the user profile ontology. After reaching this stable technical concept for the first demonstrator, it was informally evaluated based on all scenarios, indicating feasibility and possibility of improvements. This step is presented in section 5. Finally, the subsequent implementation for the first demonstrator is briefly discussed in Section 4.

In the future the technical concept will be extended to a solution for all scenarios that are selected for demonstration, leading to a generic context management solution for MyUI. The resulting solution is going to be evaluated in the upcoming user tests. Results of the user tests will continually improve the technical concepts based on real-life data; leading the way to a final solution.

As agreed at the VUMS cluster meeting in Prague in November 2010, a common glossary of terms should be provided by all cluster projects within the single projects' M12 deliverables related to user modelling. In case of MyUI this is D1.1; hence the list of terms to be defined in the glossary is attached to this document as Appendix B.

2. Context Management Approach

As already stated in section 1, MyUI aims at providing support for user interfaces that dynamically adapt to the needs of the user and the characteristics of the user and the user's environment i.e., the **context** of a user. More specifically, context can be defined as

any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.[DeyContext]

The role of context management in adaptive systems is best understood by referring to the general model of adaptive system described in D2.1 based on the work of (Weibelzahl et al, 2004; summarizing Totterdell & Rautenbach, 1990; Oppermann, 1994 and Jameson, 2001). The model describes the three typical stages involved when providing adaptive behavior:

1. Afference – collection of observational data about the user.
2. Inference – creating or updating a user model based on that data.
3. Efference – deciding how to adapt the system behavior.

Typically, context management integrates the stages of afference and inference with the goal of providing a useful representation of contextual information that enables efference. Seen from the viewpoint of context management, we can characterize afference as providing a shared sensor data repository where sensor events are collected. On top of this, inference is the integration and management of different context augmentation services. The goal of augmentation is to gather information about the user. This information must be useful to enable adaptive behavior in the phase of afference. Information about the user is structured according to the user model and stored in the user profile. While the user model describes the information collected about users in terms of a schema, the user profile is the instantiation of the user model for a concrete user. Both, the user model and the user profile, will be discussed in the course of the subsequent sections. The next three subsections will also further examine the role of context management in the light of this 3 stage model.

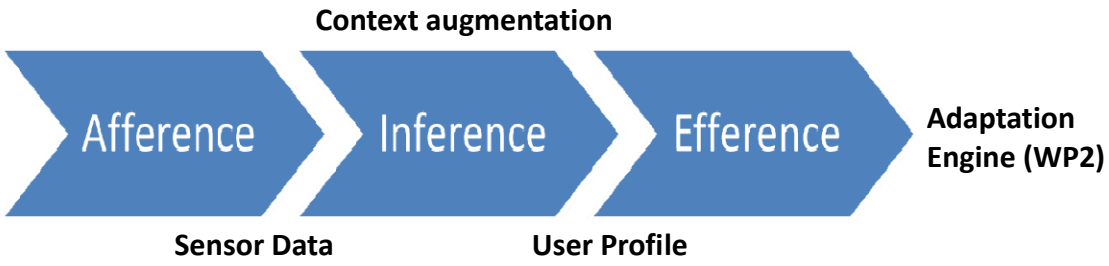


Figure 1: Approach to an adaptive system in the light of context management

2.1 Collection of Sensor Information

Looking at the problem of afference in more detail, we can distinguish implicit afference, i.e. collecting data in the background without direct involvement of the user, and explicit afference, i.e. actively involving the user in data collection. Since unobtrusive collection of user-related data without influencing the user in his or her normal interaction with the system cannot be discarded as a requirement in MyUI, the architecture of the context management solution has to account for both cases.

Implicit afference can be achieved by using sensors to unobtrusively detect observational data about the user. Unfortunately, individual sensors seldom deliver information that can be used directly to make useful statements about the user. Usually, sensor events must be connected, aggregated and augmented to arrive at information that can be used to populate a user profile.

This leads us to a generic context management architecture for adaptive systems where sensor events are collected in a sensor data repository. This shared data repository is filled by virtual and physical sensors that store their sensor events into the data repository. Physical sensors are pieces of hardware equipment placed in the surrounding of the user that are able to detect certain characteristics that are useful to infer user profile information. On the other hand, virtual sensors are pure software based sensors that detect information by analyzing the user's interaction with the MyUI system.

In the inference stage context augmentation algorithms are then accessing this shared sensor data repository and are trying to derive user profile information from sensor events, additional background knowledge and historic context information.

For more information on sensor modeling and the sensor data repository, please refer to Section 3.

2.2 Context Augmentation

As described in [Kobsa 2004] rule-based approaches, probalistic reasoning, data mining, predictive reasoning, and other machine-learning approaches might be used to derive useful information about the user. These different approaches each have different drawbacks and benefits, e.g. probabilistic methods might have benefits in activity recognition, whereas rule-based approaches are easier to maintain and implement. To account for different possibilities the general context management architecture should be able to incorporate different reasoning mechanisms.

This can be achieved by a data-centric integration architecture. As analyzed by Winogard in [WinogardContext], blackboards are the architecture of choice for this problem. In such blackboard architecture, sensor events and other useful information are published on a shared blackboard. Context augmentation services that represent different reasoning and inference techniques and algorithms can then access the information on the blackboard and derive useful user profile information. Compared to other approaches such architecture focuses on a separation of the shared

descriptive data that is captured in sensor data repository and user profile from the algorithmic and processual knowledge that is needed to infer additional information. Consequently, context augmentation services and therefore reasoning algorithms can be replaced independently. This enables fast prototyping, by easily experimenting with different reasoning approaches.

2.3 User Model

Capabilities and limitations of humans change over time, i.e. during the process of aging. As described in the deliverable D2.1 (section 5.2.2.1) one example is the ability of hearing, which decreases over time. Although it is common that abilities decrease during aging, there is no uniform pattern of the steady cognitive decline. Sometimes it is also possible that a loss of cognitive abilities can be reversed. This induces that the user profile is never fixed at a certain point in time. Rather the user profile has to be continuously adapted to the user it is associated to.

In MyUI context management enables adaptive user interfaces by providing a continuously updated user profile. Of course, to enable efference the user model must provide information that is able to trigger and drive adaptation. Since MyUI is dealing with the adaptation of user interfaces, MyUI's user model is focusing on impairments and disabilities with regards to human computer interaction.

For more information on the actual content of the user model please refer to Section 3. The next two subsection deal with the important interrelated technical concept of providing an up-to-date representation of the user's current context.

2.3.1 Toward an Up-to-date Representation of Current Context

Independently of the content that is concretely modelled within a user profile, the notion of the current context is an important concept of context-aware systems. Representing the current context of a user is a non-trivial task in domains where conflicting information might be available. This could be due to sensor information indicating two different values for a user model concept at the same point in time. Conflict-resolution therefore deals with the provision of a conflict free representation of the user profile based on possibly conflicting input information.

Especially, in an approach where the user profile is adapted continuously, there is a need for a strategy to prevent or diminish the effect of false updates. In the most simplistic case the measurement of two sensors could result in the update of the user profile with a positive fact, and the other sensor would measure a negative fact, nearly at the same time. In an open-architecture like MyUI, where different sensor technologies, context augmentation services, UIs and applications are supported, it is impossible to clearly prohibit the occurrences of wrong or conflicting updates to the user profile.

Other types of conflicts would be to violate cardinalities. This would be the case if there are more facts (which correspond to statements in the wording of MyUI) in a set of facts, where there are multiple occurrences of one given characteristic. In this case there would be no way to "know" the right value of the characteristic, since only one user model concept can be valid at one time. For example one could consider having multiple instances of the fact "user A has a visual acuity of B".

In any case, a method for dealing with conflicting updates is needed. The solution is to derive a conflict-free representation of the current context. This can be achieved by evaluating a user profile update as soon as it occurs (conflict-resolution on storing) or it can be done when conflict-free information is requested (conflict-resolution on query). In order to resolve these conflicts there are multiple strategies possible. Conflict resolution is done by a so called conflict resolution operator, which is a function that transforms a set of context facts into a set of conflict-free context facts. [SchmidtPhD09]

For the demonstration of the first prototype the conflict-resolution on query approach was chosen, because this allows us to keep all context information inside the internal storage longer than in the conflict-resolution on storing approach. Possible operators can include:

- ” most recent” – considering only the last update
- “average” – considering the average over a period of time
- complex combinations of both
- etc.

Which operator or combination of operators is chosen, is closely connected to the way the user profile will be updated. If irregular updates with no or few conflicts occur the most-recent strategy can be best. If updates occur in high-frequency with some conflicts or errors an approach based on averaging is suitable. The way of how the user profile is updated, is referred to as update strategy.

An update strategy describes the general approach to updating the user profile. For MyUI the update strategy can be summarized as accumulating multiple minor incremental updates, whereas every sensor event is translated into such an update by means of a context augmentation component.

Email Client Example:
The user is interacting with the email client application by using a user interface that is based on a generic design pattern The generic design pattern is described as suitable for people with mildly-limited visual acuity. The distance sensor is detecting that the user is leaning forward when interacting with the information on the screen. A software component is analysing all this information and comes to the conclusion that the user must have slightly worse than mildly-limited visual acuity. The component, therefore, slightly adapts the corresponding value in the user profile. This minute change will later on lead to the selection of different generic design pattern that, in turn, will lead to a different user interface experience.

Based on available sensor information gathered from virtual and physical sensors, it is hard to determine whether the user’s ability to interact with the system has improved. Therefore, this approach will manifest in repeated adaptation of user model concepts in the user profile, indicating a deterioration of user characteristics.

To overcome this problem there has to be a mechanism that continuously adapts the values saved in the user profile in a way that indicates an increase of the user’s interaction capabilities. One possible mechanism would be to detect not only the problems a user has with a specific user interface, but also to detect if a user has no problem or does actually very well in using a specific interface. Although this would be the preferred method it is very hard to detect whether a user has no problems with a user interface. Doing so would imply that every user uses a user interface in an equal way and with the same efficiency.

Another mechanism discussed inside the project is to have a decay mechanism periodically adapting all values stored in a user profile by a small constant value. Although, this approach is very easy to understand and implement; it will be hard to find the correct parameters for it. If the user profile is updated too often by this mechanism the user’s limitations are constantly underrated which would lead to a user interface not suitable for the user. On the other hand; an infrequent update of the user profile would neutralize the effect such a decay mechanism would provide.

Additionally to the decay mechanism it would also be possible to regularly analyse historic sensor events. I.e. if the distance- / gesture-sensor hasn’t detected a lean-forward gesture during a given timespan the system could try to decrease user profile variables again.

Currently there is no implementation or concretization of such a mechanism included in MyUI. This is the case, because the overall plan of WP1 has shifted from a more analytic approach where

algorithms are designed based on analysis of existing knowledge and user studies towards a rapid-protyping approach in which we develop a running system as a basis for experimenting and gathering runtime experiences. Within year 2 of myUI, WP1 will develop further concrete heuristics for deriving user context information based on the data gathered from sensors and expected results.

3. Context Ontology

User context is the system-side representation of a user's situation as far as it is *relevant* to the system, framework or application at hand (see [SchmidtContext]). In MyUI, context information is used to drive the automatic adaptation of user interfaces, i.e., it is focussed on those aspects of a situations that (a) can be captured with the help of sensors and (b) are useful for deciding upon adaptation strategies. Compared to other projects in VUMS cluster, most notably the GUIDE project, this leads to a different approach in designing a user context ontology. In GUIDE, the ontology (called "user model") is aimed at a description of the user in a form that is as realistic and complete as possible and enables use of the model for simulation of user experience. While in theory such an approach is very powerful, it inevitably introduces a level of complexity when it comes to runtime instantiation of such a model for a concrete user, based on a concrete set of sensors, and for a concrete set of adaptation strategies. The amount of heuristics required typically makes testing and fine-tuning of such a system very challenging.

Within MyUI, we have therefore deliberately focused on those aspects that "make a difference" when it comes to adaptation strategies. The most promising approach was based on modelling disabilities for which UI designers develop patterns, based on which we select appropriate patterns at runtime, and which we can derive from combining sensors and user interaction data or can easily get static input.

3.1 Method

Following the state of the art in computer science, an ontology is "a formal, explicit specification of a shared conceptualization [StuderOntology]. Ontologies are used to derive data structures and schemas as well as software interfaces providing access to information represented in those schemas. Developing a "good" ontology is always an iterative process in which we have to find out the appropriate representation, which is formal enough to enable automated actions, useful for describing the problem at hand, and which represents a shared understanding of those using the ontology. Therefore we have adapted the approach illustrated in fig. 1 which further described in [SchmidtPhD09] to the needs of the MyUI project. The given method has already been used in other EU-projects like LIP (Learning In Process), Agent-DYSL and SOPRANO.

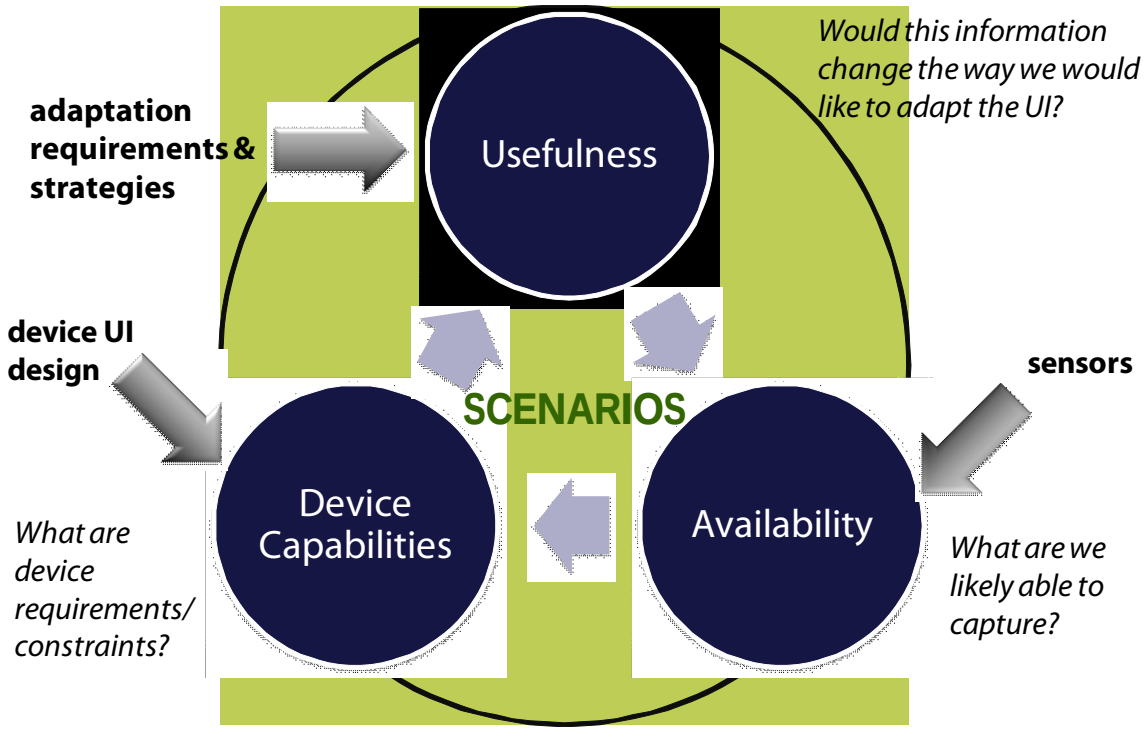


Figure 2: Iterative development of the MyUI context ontology

Within MyUI usefulness of user profile was mainly based on input of D2.1 and an initial evaluation of the project scenario as described D4.1 leading to an initial set of user disabilities and impairments in relation to human computer interaction. In several discussions with different consortium members Availability and Device Capabilities have been checked leading to a iterative refinement of the initial list of valuable user characteristics. These steps have been repeatedly followed as part of several project discussions guided by a conceptual evaluation based on the MyUI scenarios (see Section 4 for more information).

This has resulted in identifying three distinct parts of the context ontology:

- the **user profile ontology**, describing properties of an end-user of the MyUI system
- the **sensor ontology**, describing
- and **application-specific ontologies**

These three areas are further described in the following sections

3.2 User Profile Ontology

3.2.1 Structure

In general, a MyUI user profile is collection of information about an end-user of the MyUI system. This includes personal information like email address, first name, last name, etc. as well as information about user capabilities and characteristics as far as they are relevant to determine the human computer interaction (HCI) abilities of a user. The general idea is to collect and continuously adapt HCI-relevant user information, so that human computer interfaces (also called user interfaces) can be dynamically adapted to the current capabilities, needs and limitations of a user.

In more technical terms and based on the RDF data model. RDF describes relationships in terms of resources. Resources can be differentiated into subjects, predicates, objects and statements. A statement is a triple consisting of subject, predicate and object. An RDF-statement roughly translates to a data representation of a simple natural language sentence that captures statements

about the subjects. In RDF resources are referenced by Uniform Resource Identifiers URIs. In relation to MyUI a statement always addresses a user capability, characteristic or short property of a user. The user is the subject. The property is the predicate. The value that is stated for this property is the object of the statement.

Based on this a MyUI user profile is a collection of statements referring to the same user URI, i.e., one user profile is always connected to one and only one user. The allowed properties and possible values are defined in the MyUI User Profile Ontology, which describes the constraints under which a user profile is considered valid. Only valid user profiles can be processed by the MyUI system. Two constraints have already been defined in the previous section, namely:

- C1: A MyUI user profile must be a collection of statements.

- C2: A MyUI user profile must consist of statements referring to a specific user URI. One user URI must always uniquely refer to one specific user.

When it comes to adaptation (WP2), we are using the term “user profile variable”. This corresponds to a statement where the subject is the specific user this variable belongs to. The property, which can be seen as the predicate in a RDF-statement, is also called “user profile variable name”, and the object is called “user profile variable value”.

- C3: Every property in a MyUI user profile must correspond to a specific characteristic of the user, which is called “user profile variable name”.

In short, one could say that the property must be the name of a user profile variable. A detailed overview of the currently defined user profile variables and a description of the method used to define these variables is provided in chapter 3.2.2. This list is not fixed yet which means the user profile variables (and therefore the properties) are not fixed yet and might change in the future of the project.

Although MyUI’s modelling does not rely on OWL conceptual it is closely linked to the GUMO modelling principles described in [HeckermannGumo] as

$$\texttt{subject}\ \{\ \textit{UserModelDimension}\ \}\ \texttt{object}$$

In contrast to GUMO, MyUI is aiming for a much more task-oriented user profile, where information capturing disabilities and impairments in relation to human computer interaction are captured. This makes it impossible to simply reuse the high-level user model dimensions proposed by GUMO. Therefore, MyUI is extending the GUMO model with the user variable concept.

The range of a property defines which values can be used as an object in a statement (i.e. it defines the type of the corresponding user profile variable). We can distinguish the following three cases:

- *Numerical Literal:* We use numerical literals to represent user profile variables which are ratio-scale. In this case one value is selected from a predefined, continuous interval for a user profile variable of this type. In the current ontology this interval is considered to be the same for all ratio-scaled user profile variables (in contrast to nominal-scaled user profile variables where a different set of possible values is defined for each user profile variable). The interval is specified to be [0,4] (i.e. all possible values between 0 and 4 including 0 and 4 can be assigned). Although, we want to avoid an exact medical definition, 0 can roughly be mapped to “not limited or normal”, 2 can roughly be mapped to “mildly limited” and 4 can roughly be mapped to “severely limited”. This kind of type is typically assigned to user profile variables that represent limitations and capabilities of the

user like hand precision, visual acuity. Assigning a value of 0 to hand-precision, therefore, maps to the statement that this user's hand-precision is not limited or normal. The default value is 0. However the above mapping exists, this doesn't have to be applicable for all user profile variables. The user profile variable "ambient light" for example uses another mapping (0 translates to no ambient light meaning absolute darkness, 4 translates to very high ambient light). Concluding it can be said that the meaning of the values of a numerical literal depends on the concrete user profile variable. The mapping for each defined user profile variable can be found in Table 1. Using a continuous, ratio-scaled interval has the advantage that there exists a strong order between the distinct values for a user profile variable. Furthermore using continuous values is helpful by providing the possibility to have a fine-grained differentiation.

- *String Literal:* This corresponds to a free-text user profile variable. This means any value can be assigned to this kind of user profile variable as long as it is a string. This user profile variable type is used for user characteristics like first name, last name and email address. Although, in principle any value is accepted (as opposed to a set or interval of values), the format of the string might be constrained in a clearly defined way. For example, the email address must be a syntactically correct email-address; or the first name must not exceed a certain number of letters. The default value for a user profile variable of this kind is an empty string.
- *Enumeration:* This means a subset from a set of predefined values (represented as concept instances) can be assigned to a user profile variable which is nominal-scaled. This user profile variable type is used to specify the languages that are accepted by a person. In a collection of statements, an "assigned subset" maps to a set of statements with the corresponding property for that user profile variable. The collection must not be empty if it is explicitly assigned. Enumerations also support default values, which are used when there is no corresponding statement. The default value is one specifically marked value of the set of possible values (since only subsets can be assigned, technically it would be a set containing only this single element).

All this can be condensed into the following constraints.

C4: A property with a string literal range must appear at most once. Its value can be any string that complies with the format restriction of the corresponding user profile variable. The default value is an empty string.

C5: A property that corresponds to a nominal-scaled variable can appear multiple times. Its value must be selected from the instances of an assigned range concept. If there is no property instance, the specified default value is used.

C6: A property that corresponds to a ratio-scaled user profile variable must appear at most once. Its value must be selected from an (currently common) interval. The default value is 0.

The resulting defined ontology formalism can be seen as RDF plus a subset of RDFS (concept and property hierarchies) which is interpreted as constraints, i.e. range definitions constrain the allowed values for certain properties.

3.2.2 Properties and User Profile Variables

In MyUI, a function-based user modelling approach is used. Hence, user profile variables are connected to specific user interaction abilities and constraints subdivided into perceptual, cognitive and motor attributes. Variables relevant for MyUI have been initially selected from the WHO ICF

guidelines. Further sources, e.g. the ISO 22411 standard and major requirements determined in D2.1 (Requirements for User Interface Adaptation), are also considered.

The following key questions have been underlain the selection process:
1. Does a certain attribute affect the interaction with ICT products?
2. Can user interface adaptation overcome or weaken the interaction constraints?

If both questions can be answered with "yes" the attribute has been included as additional user profile variable to the MyUI user model.

The initial list of user profile variables used in the current version of the MyUI system is presented below. Each variable is shortly explained, appropriate valid value ranges and references to existing standards are given. Possible methods of resolution are not detailed here, but will be part of D2.2 (Adaptation concept and Multimodal User Interface Patterns Repository).

Area	User Profile Variable	Interpretation	Range	Reference
Perceptual Vision	visual acuity and sensitivity	ability to perceive what is displayed on the screen	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b2100 visual acuity functions, b21022 contrast sensitivity, b21020 light sensitivity ISO 22411 URS07, URS09 FRS01-02
	colour perception	ability to distinguish colours (without any limitation such as e.g. red-green, blue-yellow blindness)	0 - colours can be distinguished without restrictions 1 – not all colours can be distinguished	WHO ICF b21021 colour vision ISO 22411 URS15 FRS02
	field of vision	ability to perceive without limited vision in certain areas	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b2101 visual field functions ISO 22411 URS10 FRS02
	ambient light	ambient light conditions	[0, 4]; 0 - no ambient light, absolute darkness; 4 - very high ambient light level, e.g. dazzling sun light on the screen	WHO ICF b21020 light sensitivity ISO 22411 ER01-03 FRS03 FRE01
Hearing	hearing	ability to hear	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b2300 sound detection ISO 22411 URS11

	ambient noise	ambient noise conditions	[0, 4]; 0 - normal; 4 - high noise level	WHO ICF b2300 sound detection ISO 22411 FRE02
Cognitive Language	language reception	ability to comprehend written or spoken language	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b1670 reception of language ISO 22411
	language production	ability to produce written or spoken language	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b1671 expression of language ISO 22411 URC07 (limited literacy)
	understanding abstract signs	ability to understand abstract symbols or icons	[0, 4] 0 - normal; 4 - severely impaired	URC20
	attention	Covering selective, focused and divided attention abilities[1]	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b140 attention functions ISO 22411 URC05
Information processing	processing speed	ability to process information in an appropriate time	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b1470 psychomotor control ISO 22411 URC04 FRC01
Memory	working memory	ability to remember exact sequencing of multi-step procedures and to orientate oneself in long sequences of steps	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b1440 Short-term memory ISO 22411 URC03 FRC02
	long term memory	ability of learning, storing and retrieving new information	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b1441 long-term memory ISO 22411 URC02 FRC02
ICT skills	ICT literacy	skills and experiences in using current ICT user interfaces	[0, 4] 0 - normal; 4 - severely limited	URC01

[1] selective attention: ability to attend to one thing without losing control of another one
focused attention: ability to concentrate on only one thing at a time and get not distracted by irrelevant information
divided attention: ability to split attention to more than one thing at a time

	need for security / ICT anxiousness	importance to which security is needed including risk-avoiding	[0, 4] 0 - normal; 4 – particularly pronounced	URG03 FRG10/11
	hand-eye coordination	ability to move hands according to visual feedback	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b1471 Quality of psychomotor functions ISO 22411 URC 19
Motor Speech	Speech articulation	ability to articulate speech	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b310 voice functions, b320 articulation functions ISO 22411 URP07 FRP04
	finger precision	ability to move the fingers	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF d440 fine hand use ISO 22411 URP02 FRP01, FRP03
Dexterity	hand precision	ability to move the hands	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b730 muscle power functions WHO ICF d4453 Turning or twisting the hands or arms WHO ICF d4402 Manipulating ISO 22411 URP10 FRP01-02
	arm precision	ability to move the arms	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF d4453 Turning or twisting the hands or arms
	contact grip	ability to apply a unidirectional force by a finger, the thumb or the hand [according to ISO 22411], e.g. by touching or operating via touching	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF b265 Touch function ISO 22411 URP06 FRP03
Muscle strength	pinch grip	ability to hold the control by the fingers and/or thumb without clenching the fist hand [according to ISO 22411], e.g. when holding sth between the fingers.	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF d440 fine hand use, d4400 Picking up ISO 22411 URP06 FRP03

	clench grip	ability to use all fingers wrapped around a control [according to ISO 22411].	[0, 4] 0 - normal; 4 - severely impaired	WHO ICF d4402 Manipulating ISO 22411 URP06 FRP03
General data	first name	first name	any string	
	last name	last name	any string	
	email address	email address	any well-formed email address	
	preferred language	preferred languages	**English**, German, Spanish	URG02 FRC04

Table 1: Already defined MyUI user profile variables. Default values are written in bold.

3.2.3 The User Profile as basis for the individual user interface generation

Generic design patterns refer to certain user profile variables stored in the user profile in order to set global user interface features as font size. In this way, an individual user interface profile is created. The individual user interface is composed by different user interface elements. Each user interface element is described by a related interaction design pattern which in turn is based on specific global variables from the user interface profile and the current interaction situation.

In general, design patterns (generic and interaction-) are selected if their preconditions (rules that define tolerance intervals or tolerated values for the relevant user profile- and global variables respectively) are fulfilled. The preconditions are illustrated in the following on the example of generic design patterns.

As can be seen form the user profile variables table in the previous section, string-based user profile variables only appear in the general data part of the user profile – which can be considered as static during the interface adaptation process.
The majority of the remaining user profile variables are ratio-scaled, some are nominal-scaled. For these two data types, examples are presented for simple preconditions:

- *Ratio-scaled user profile variables:*
  A simple precondition of a generic design pattern could define an interval restricting allowed values for a certain ratio-scaled user profile variable. If a user's individual assignment of this ratio-scaled user profile variable is included in the given interval, then the precondition is met and the generic design pattern selected.
  For example, considering a pattern suitable for people having normal or slightly limited hand precision. So, the precondition is "IF 0<= hand_precision <=2". In consequence, for all users that have a value less-than or equal-to 2 assigned to this user profile variable this generic design pattern is applied.
- *Nominal-scaled user profile variables:* here a simple precondition might check the presence or absence of a specific user profile variable. If a user's individual assignment of this user profile variable equals the hypothesis, the generic design pattern is selected.

Concerning the interaction design patterns, similar preconditions are used. The only difference is that they refer to global user interface profile variables and a specific interaction situation.

3.3 Sensor Ontology

The MyUI framework should enable dynamic user interfaces that adapt to the changing capabilities of end users. This dynamic behaviour is based on a dynamic user profile that is continuously updated to reflect the current status of the user. To efficiently and unintrusively sense the current status of the user, sensors need to be installed in the direct environment of the user. Sensors can detect and measure information about the user. This information can be used to derive the current situation of the user.

3.3.1 Structure

As explained in the previous section the MyUI context mainly consists of information about user capabilities, limitations and characteristics as far as they are relevant to determine the human computer interaction (HCI) abilities of a user. This kind of context information must be derived from sensor data.

In MyUI, sensors are roughly be mapped into two categories:

- physical sensors that require a piece of hardware and software and
- virtual sensor that just analyse and report the interaction of the user with the system.

Both types can be modelled in the same simple fashion.

The key modelling approach is that a sensors measures sensor events. Sensor events usually are modelled as triples composed of a sensor URI, a property defining what has been measured and the value that has been measured for the property. This approach of effectively modelling sensor states has been derived from the state based modelling shown in [BoninoDogOnt].

Sensor Event = (sensor id, measured property, measured value for that property)

The MyUI sensor modelling is limited to this simple modelling approach to allow for flexibility and easy extensibility. As with the list user profile variables also the list of measurable properties and the values that can be measured for a certain property is not fixed yet and most properly is to be extended when more and more scenarios are implemented. Figure 2 shows an abstract graphical overview of the sensor ontology.

Currently the project integrated a RFID reader for user identification, an IR-sensor for the remote control, and a gesture-/distance sensor (based on a web-camera). These physical sensors are modelled as sensor objects. To allow measurements the following measureable properties have been defined:

- RFID-tag In Range
  This property states that the RFID reader detected a transponder. The value associated with this property contains the identifier of the detected transponder.
- Remote Control Keypress
  This property states that a button on the remote control has been pressed. The associated value contains the code or identifier of the pressed button.
- Head To Display Distance
  During the usage of the MyUI system, the distance and gesture sensor continuously measures the distance between the head of the user and the camera which is normally the distance between the head of the user and the display. Measurements are periodically reported as sensor events with the “Head To Display Distance”-property. The associated value states the absolute distance in centimetres.
- Gesture
  The sensor measuring the distance between head and camera also continuously analyses the user’s movement to determine special gestures. Two gestures currently implemented

are the lean-forward, respectively lean-backward gesture. This property states that one of these two gestures has been detected by the sensor and the value associated with the property contains either “forward” or “backward”.

How these properties are used to alter the user profile is described in chapter 4.

To enable automatic augmentation based on sensor events, background knowledge must be provided for the augmentation process. Consider, for example, a RFID reader that is detecting whether a transponder is in range or not. Although this information itself is not very significant, it is possible to derive further information if additional information is present. For example it would be possible to derive that a specific person entered a room if there are two additional statements:

1. The detected transponder is connected / carried by a given person A
2. The RFID reader is mounted at a specific room B.

With this information one can derive that person A entered room B. Facts like the two abovementioned facts about the RFID reader are called **meta-statements**. These meta-statements indicate how the sensor is connected to its environment which, in turn, defines how sensor events can be interpreted. To allow for reasoning over sensor information this meta-information must also be captured as part of the user profile. In particular, it must be defined when a new sensor with its sensor id is incorporated into the system. In general, such meta-information must be known about sensors and other physical devices in order to make sense of the sensor events that the sensors deliver. Additionally, though, it might also be useful to specify such information about other things that are not sensors, e.g. physical objects that participate in a sensor measurement like an RFID-tag. Therefore, the concept of a physical object was created to indicate things that might require specific meta-information, in order to make sense out of them.

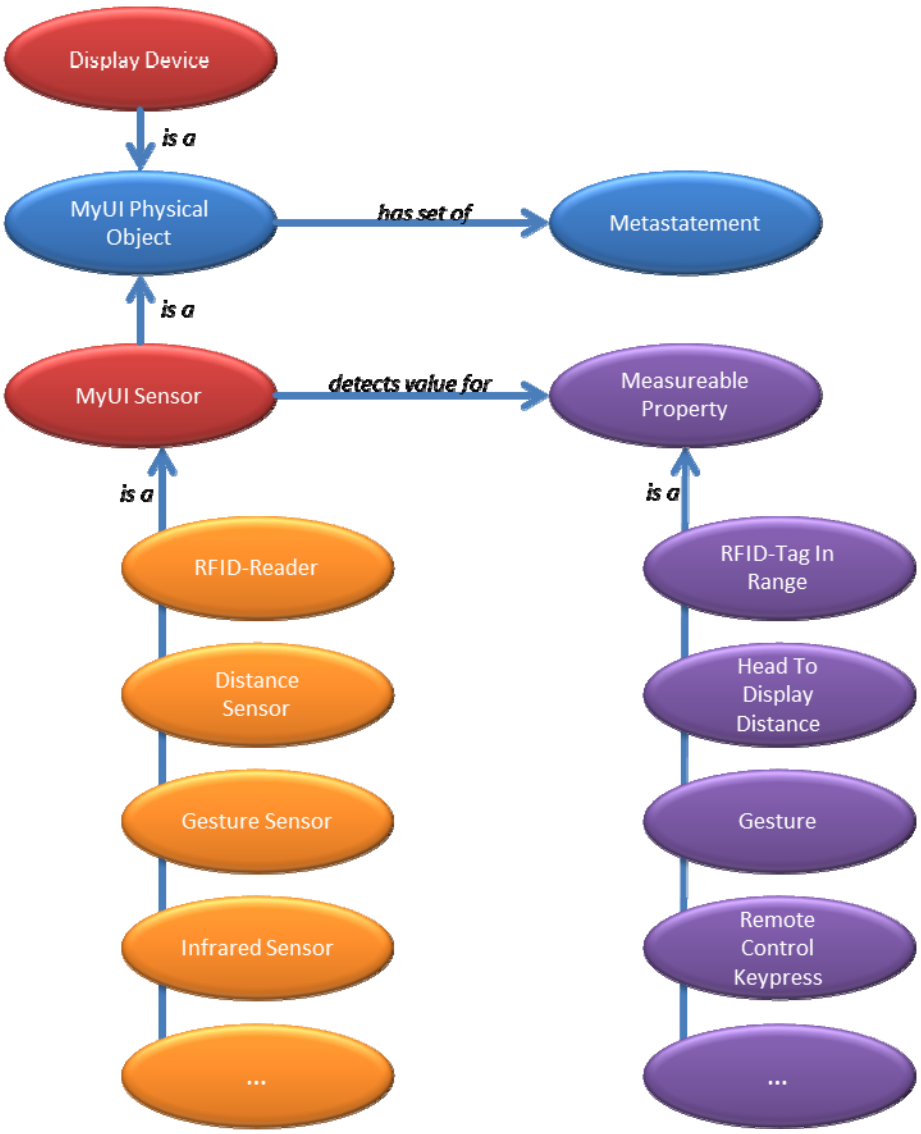


Figure 3: The MyUI Sensor Ontology; purple colours indicate measureable properties, orange colours indicate concrete sensors, red colours indicate abstract physical objects, and blue colours indicate the highest level of abstraction.

3.3.2 Example Modelling of MyUI sensors

Since sensor information is concrete and not abstract information, it is very likely that the list of sensors and measurable properties needs to be extended regularly when new applications and scenarios are implemented (in contrast to the user profile whose list of user profile variables is abstract and will therefore most likely converge into a fixed list over time). This example should therefore guide future MyUI system administrators when extending the sensor ontology.

Distance Sensor Example:
As it has already been mentioned in section 3.3.1 the distance sensor detects the distance between the user's head and the display that is used for interaction. Hardware wise the sensor is based on a camera system, software wise the sensor will deliver information about the distance in centimetres. This information could be easily mapped into sensor events like this:

Distance Sensor Event = ("Some distance sensor id", "Distance to head", "60")

In this example, the MyUI framework would know that sensor id “Some distance sensor id” maps to a distance sensor and that the “Distance to head” property requests a numerical value that indicates the distance in centimetre. In order to derive useful information from this sensor event, a distance sensor must also be connected to some display via a meta-statement. Furthermore, either the display or the distance sensor must be connected to the user whose head is indicated in the measurement. Only with this additional meta-information, the system is able to infer information about the distance of a user’s head to a real display.

For various reasons it might be beneficial to abstract the sensor event as much as possible. For example, MyUI is using the distance sensor to detect the leaning forward and backward gesture of a person interacting with a display. Therefore, a more suitable sensor event representation might be:

Distance Sensor Event = (“Some distance sensor id”, “Lean-forward or lean-backward gesture detected”, “forward” (or “backward”))

In this case the measurable property “Lean-forward or lean-backward gesture detected” is requesting one of two possible values: “forward” or “backward”. One obvious benefit of this type of modelling to one presented before is, that much less data needs to be exchanged between the sensor and the sensor data manager, since there is no continuous stream of distance sensor events like they have been introduced at first. The “Lean-forward or lean-backward gesture detected”-event would only be triggered if the user performed either a lean-forward or a lean-backward gesture.

3.4 Application Specific Data

In addition to storing sensor and user profile specific information, the analysis of the scenarios also indicated a need for storing application specific data. Applications provide a service to the end user. In MyUI, applications only specify the general outline of user interaction by specifying interaction situations and deal with “content” that is handled by the application; the creation of user interfaces and other tasks have been shifted into other components of the MyUI framework, with the goal of making application development easier.

The analysis of the scenarios and the underlying applications indicated that every application needs to store at least some application specific data. For the MyUI email client application this could be login information for the underlying email server, for the physiotherapy application this could be more complex information about individual exercises or reference to exercise videos. Since the need for an application specific storage capability was so ubiquitous, it was decided to add the functionality for storing application specific data to the context manager; and, therefore, simplifying the burden of the application developer even further.

The concrete ontology for application specific data has to be defined by the application developer; since needs of applications cannot be foreseen in an architecture that is open for new applications. As with the other two parts of the context manager the general modelling paradigm is based on a key-value pair approach. This means one data entry, specific to one application and one user, is modelled as a collection of key-value pairs.

Application Data Entry = (user id, application id, collection of key-value pairs)

For the above mentioned email client application the entry could look like this:

Email Client Data Entry = (“some user id”, “email client application id”, (“gmail user name” = “peter”, “gmail password” = “peter”))

Any client software that knows the user id and the application id can access this application specific collection of key value pairs. An application developer can simply add a new collection of key-value pairs for an individual user by specifying the user id and the id of the application.

4. Context management implementation

4.1 Description of the components

The MyUI context management consists of three components: the Sensor Data Manager, the User Profile Manager and the Application Data Manager. Their interfaces will be exposed to the other subsystems of the MyUI architecture via XML-RPC. This accounts for interoperability between the different components and allows for distribution over multiple connected computers (e.g. via the internet).

As discussed in chapter 3.4 the MyUI framework provides applications running on top of it with the ability to manage data in an application-specific ontology. This is accomplished by the implementation of the Application Data Manager that provides methods for getting, setting and deleting application-specific data.

Furthermore there exists the User Profile Manager. This module handles storage and updating of the user profile variables for every user. (see chapter 3.2). To update a user profile variable, other components from inside and outside the Context Management have to provide a valid – in terms of the user profile ontology – statement about the user (e.g. “Arthur has-visual-acuity-and-sensitivity 2.3”). Updates can be provided either directly by external components or from internal components like the Sensor Data Manager. With this architecture it is also possible to account for virtual sensors, where applications provide feedback about the user’s behaviour while navigating through the application and update the user profile accordingly. For example an application can detect if the user navigates through the application in loops, which could indicate that the user has a severely impaired working memory and update the user profile variable of the user directly via the User Profile Manager.

Besides storing and updating the user profile the User Profile Manager keeps track about the history of the user profile and provides the ability to retrieve a complete list of user profile variables with their respective values for one user.

The third component is the Sensor Data Manager, which is responsible for the transformation of sensor events into user profile statements as mentioned in chapter 3, and calling the User Profile Manager to update a given user profile. In order to do this it exposes a method publishSensorEvent via XML-RPC that other components in the MyUI system can call. The Sensor Data Manager receives its input from the Event Processor which publishes for example a detected lean-forward gesture. The transformation of sensor events into updates of the user profile is done rule-based, but more advanced techniques can be employed in a later stage of the project.

All components mentioned beforehand have access to an object-oriented storage which is used in order to store all kinds of statements and which can be queried by all components inside the Context Management. This accounts for the need to persist context information and the usage of past context data during the analysis of the context.

4.2 OpenAAL

The implementation of the MyUI context manager is based on the open-source software openAAL[2] (see also [WolfopenAAL]). openAAL has been developed within the European-funded project

[2] www.openaal.org

SOPRANO[3] (see also [WolfSoprano]). It is supported by an active community and has been released in several versions since its initial release. openAAL is currently used in the following projects: FZI Living Lab AAL[4], Mirror [5]and EasyCare[6]. Furthermore, it is considered as input project into the standardization activities of the European-funded project universAAL[7].

OpenAAL has been chosen because it was originally designed with a focus on context management for applications similar to the ones to be achieved in MyUI. Since OpenAAL also utilizes the methodical workflow afference, inference, efference, introduced in chapter 2, the adaptation costs to the requirements of MyUI are minimal. Furthermore the rapid prototyping approach of MyUI applications is enabled, since FZI has as the main contributor of OpenAAL has a lot of experience on the adaption of this software.

4.3 Extensions to OpenAAL

One major adaptation to be applied to openAAL with regards to the requirements of the MyUI project is the addition of web service capabilities to the framework. That is, MyUI specific user profile-, application data- and sensor-manager interfaces need to be published as web services. Due to ease-of-use aspects and ubiquitous availability of client libraries for different programming languages (java, c++ and php) and operating systems, XML-RPC was chosen as web service protocol.

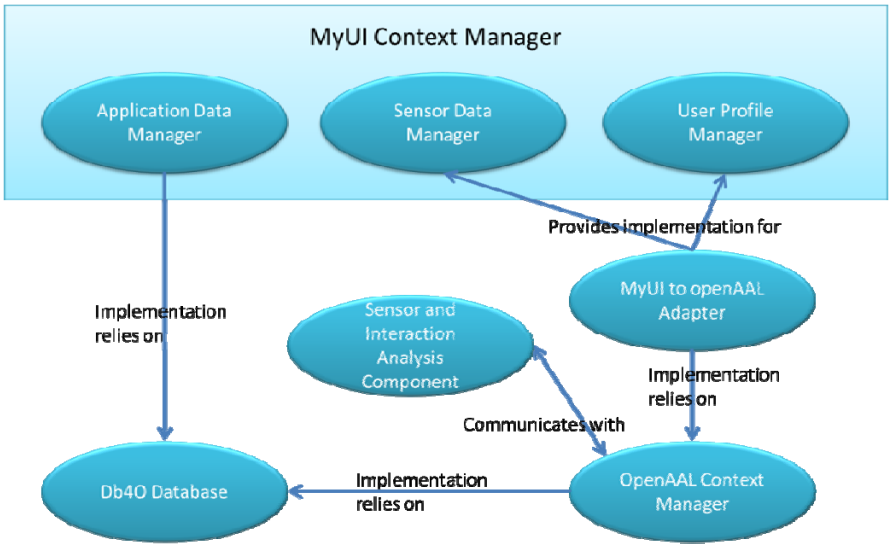


Figure 4: Context Manager Component Architecture

As one can see in Figure 3, the application data manager is directly implemented on top of the Db4O[8] database framework. Db4O provides an easy-to-use and scalable object-oriented database and has been chosen because it was also used in the implementation of the openAAL framework. The sensor and interaction analysis component is an internal part of the context manager. As exemplified in the scenario section, its major purpose is to interpret incoming sensor events in relation to existing background information with the goal of inferring meaningful updates, referred to as context augmentation, to the user profile. It is based upon openAAL's uplifter framework which provides support for simple Event-Condition-Action rules (see [KresserReasoning]).

[3] www.soprano-ip.org
[4] Aal.fzi.de (only German)
[5] www.mirror-project.eu
[6] www.projekt-easycare.de (only German)
[7] www.universaal.com
[8] www.db4o.com

5. Realisation of MyUI scenarios

The following sections exemplify the use of the MyUI context manager with the help of the MyUI demonstrator scenarios. The next section provides a step by step walk through of the email client scenario. This scenario has been selected for the first demonstrator and is, therefore, treated with special care. The thereon following section provides a short overview about relevant concerns taken from all the other scenarios that are presented in D4.1 and shows how they can be handled with the context management approach presented in this deliverable.

5.1 Email Client Scenario

This section provides a step by step email client scenario walkthrough. After each step the realization with MyUI's context manager approach is indicated. The scenario text was taken from the email client scenario stories document that can be found as Appendix 1. It only depicts a very simple adaptation case based on one user profile variable and two type of sensor information; just to exemplify the general approach.

5.1.1 Scenario Walkthrough

"Arthur wants to use his new Net TV to see if his daughter has sent him an email. He starts the Net TV by pressing the „on"-Button on his remote of which he knows the position. Arthur selects the email service by choosing the relevant button on the screen by pressing the according number on the remote. The screen shows 3 Messages with the name of the sender and the subject of the mail in big letters …"

> When selecting the email service the email application is started. The email application is asking the context manager for the current profile of the user. For this purpose a user id must be communicated to the application when the application is requested. This id is used to request the user profile from the context manager. In the scenario, the user profile does already contain non-default values for some of its user profile variables; probably because it was initialized in that way after a user examination. Especially the user profile variable "visual acuity" does contain a non-default value of 2.5, indicating that the person has limited vision. The value of the user characteristic "visual acuity" now gets translated into an appropriate value of the global variable "font size" by a generic design pattern. In case of Arthur medium font size is set. Based on the global setting for font size (and possibly other global variables settings) and the current interaction situation (displaying the listing of received emails) the best-fitting generic design pattern is selected from the bundle of interaction design patterns offering different presentation styles for the list of received emails. For Arthur 3 emails are listed on the screen with senders' names and the email's subject presented in medium font size.

"… but Arthur cannot read both of the texts. He tries to get nearer to the Net TV (leans forward) to see if he might be able to read the text from a lesser distance. The Net TV recognizes his movement … "

> Arthur cannot read the text and leans forward. This forward movement is detected by a sensor and the information is sent to the sensor data manager by an event processor translating the raw events received from the sensor into statements the sensor data manager can handle. Recognizing the new sensor event, the sensor data manager tries to aggregate and combine sensor information with other available information to make meaningful updates to the user profile.
> In this case, the component is accessing the currently selected interaction design pattern checking for underlying user profile variables that are connected to the lean-forward gesture (by means of the context augmentation services. It reasons that "visual acuity" is the major underlying user characteristic and examines the related maximally tolerated

value of 3 (access is provided by the interaction design pattern). Based on this information the context augmentation service concludes that the visual acuity of the user must actually be worse than the value of 3 derived from the pattern. Accordingly, the component updates the user profile to a value of 3.1 for visual acuity. Therefore, the user profile variable is updated from a value of 2.5 to a value of 3.1.
Note that the delta by which the variable is increased needs to be adapted according experience gathered in user trials. Also, this example is only covering a very simple inference step based on one single sensor event.

"… and increases the font size."

The user profile has been updated to a value of 3.1 for visual acuity. The generic design pattern that was selected before had a maximally tolerated value of 3.0 for visual acuity. Consequently, this pattern does not fit the user profile anymore; a new generic design pattern from the bundle of generic design patterns for font-size is selected. Hence, also another interaction design pattern gets active presenting the list of received emails in a different layout. When the page is reloaded, the subjects and senders of the 3 listed emails are presented in an even bigger font size.
Note, that an immediate change of the font size is hard to achieve when relying on web technologies. User tests must show if a regular page reload every few seconds or minutes is tolerated or whether changes are only accepted when the application is requested anew.

"Still, Arthur cannot read the text so he tries to decrease the distance again. The Net TV sees that it does not help to increase the font size any further and switches to pictures of the sender and a button to open the mail."

It is recognized that Arthur still is not able to read the text which leads to an image only presentation mode as a further increase in font size is not supported by the system.
Note that the decision to switch into an image only mode is not necessarily computed on the Net TV as indicated in the scenario description. Anyway, the mechanics work in the same way as above, only that this time the sensor data manager component also has access to past sensor events and past conclusions in relation to the current user. After an inference step similar to the one described above the user profile variable for visual acuity is updated again to a value of 3.5 indicating a severe impairment of visual acuity. When the page is reloaded the layout will be changing again presenting all the information in an image only mode.

"Arthur recognizes one of the pictures as his daughter and presses the according button on the remote. The Net TV switches to the message screen where the text as well as a button to read the message and different reply buttons are shown. The message is read by the system (Text2Speech) automatically. After the system has read him his daughter's message he also describes the options available to Arthur (audio menu)."

New or subsequent user interfaces are also created from interaction design patterns that match the current interaction situation. Depending on the global user interface variables set by the individual user profile entries, a certain variant from the bundle of interaction design patterns foreseen for each new interaction situation is selected. Therefore, if the user profile settings and with it the global user interface variables stay unchanged, the variants chosen in different interaction situations will be based on the same interaction modes (e.g. solely presenting text or using images or audio output).
Accordingly, text-2-speech is activated by the interaction design pattern that is selected for presenting the content of the email. (Any other pattern does not fit to a value of 3.5 for visual acuity.)

"His daughter asks him if a pickup for grocery shopping next Monday, 4 pm is okay for him, but since he has visitors at the time; he presses the button "record message" to record a reply to his daughter. The Net TV tells him that he can start to record after the „beep"; and also tells him how to stop the recording when he is finished."

> Similar to the explanation above speech recognition is activated by the generic design pattern that is selected for answering the email.

"Arthur waits for the „beep" and starts to speak his message. After the email is sent, a written dialogue pops up to ask Arthur if the changes made to the interface while using the Net TV were acceptable to him."

> Obviously, the self-evaluation interface that pops up at the end is based in a different self-evaluation application; and is not necessarily part of the email application. Every application that is serving a specific user and knows this user's id has access to the individual user profile and is able to adapt it via the interfaces to the user's profile. Considering this, the self-evaluation application should probably not present its content as text and instead better request an appropriate interface from the adaptation engine. Considering the current user profile of Arthur, this interface would probably not be based on text.

5.2 Other Scenarios

This section provides a brief overview about all the other demonstrator scenarios presented in D4.1 and describes the role of the context manager in those scenarios.
Please note that the "Reading Emails and Messages" scenario has already been discussed in the previous section and is not considered again in this section.

5.2.1 Virtual User Lab

In the Virtual User Lab scenario, a company is using MyUI Virtual User Lab to evaluate and test an application.

In relation to the user profile the scenario mentions that the company should be able to test user interfaces and application with different user profile instances. Also it should be possible to experiment with different values for different user profile variables. MyUI's open web service based architecture makes it easily possible to create and update user profiles; even by just reusing an existing context manager instance that is hosted remotely. Currently, though, only an API is provided to interact with the user profile. To further enhance usage, a web-based user interface for handling the context manager might be necessary. This will be considered in later stages of development.

5.2.2 User Adaptive Connected Television Interface

This scenario exemplifies the capabilities of an interactive television set up as part of a MyUI installation.

One aspect of the scenario, exemplifies how such a television can be used in an initial assessment of the user capabilities. This initial assessment can lead to a very reliable initial user profile. Although, this initial assessment software is not provided by the MyUI context manager, the resulting user profile can easily be instantiated and stored for the particular user for reasons mentioned above.

Additionally, the scenario indicates that two different user profiles should be available and that it must be possible to switch from one user profile to the other just by providing a different user id; through, for example, holding the users RFID card in front of the RFID-reader. Since the context

manager does not restrict the number of user profiles, this is possible. The possibility of hosting the user profile remotely on a web server makes it also possible to have one user profile that can be "used" in different MyUI installations.

5.2.3 Cognitive Games

The cognitive games scenario, in its current form, gives no rise to any specific concerns in relations to the user profile, besides those already mentioned in the sections above.

5.2.4 Supporting Physiotherapists in Customizing and Monitoring Progress of Stroke Patients

This scenario shows how the MyUI system can support physiotherapist and patients in instantiating a regular customized exercise regime.

In relation to the context manager this scenario has lot of interesting implications. Foremost it shows an application that is configured by the physiotherapist that is later on used in the home installation of the patient. Since the general MyUI architecture is designed around the idea of providing remote access, it is no problem for the physiotherapist to create a new user profile based on his initial assessment of the patient from his medical practice. The MyUI installation in the home of the patient will later on access this user profile as well. Of course, it is also always possible to host everything on one computing device in the home of the user as it is indicated in the scenario.

Additionally, the scenario indicates the need for an application data management. Since not only the user profile, but also some information on the exercise schedule are configured by the physiotherapist. The exercise application can store this information in the application data manager. Please note that the application is responsible to handle application specific data. Currently, only the responsibility for storage of that data can be shifted to the context manager; especially a user interface for accessing the data must be provided by the exercise application. Also, the application data manager does only allow for storage of key-value pairs. Storing of videos, sounds or images is not possible. Based on the we-based architecture of the system this is not considered a problem, since, for example, the exercise videos can be stored at any URL and the URL can be stored as part of the application specific data.

5.2.5 Stroke Patient Physiotherapy Reinforcement

In relation to the context manager, this scenario adds an aspect of automatic exercise recognition to the scenario analysed above. Automatically evaluating several exercises, adapting the exercise regime and individual exercises based on that evaluation and giving tailored recommendation to the user is an extremely complex and application specific task. Such extremely specific reasoning algorithms are usually highly dependent on a particular sensor infrastructure layout that is providing the data. This makes it difficult to share such extremely specific algorithms. But the open sensor data manager interfaces and the web-based design on the context manager do at least not prohibit a sharing of complex recognition algorithms. Also in contrast to other ontology languages the MyUI ontologies do not (pre)define what types of reasoning can be applied to context information (for example, to the OWL ontology language already incorporates reasoning based on a subset of the predicate calculus). This feature that is inherited from the underlying openAAL framework makes it, in principle, possible to integrate different reasoning algorithms.

5.2.6 Supporting Healthy Exercise for the Elderly and Digital Picture Frames

In relation to the context manager, these two scenarios do not offer any additional insights into the usage of the context manager.

6. Conclusion

This document presented the concepts behind the context ontology and context management architecture. Further, a detailed look on the MyUI scenarios indicates that model and architecture do indeed fit the MyUI demands. As already mentioned the results presented in this document are not complete at the moment, since the development in MyUI follows an iterative process. The focus concentrates on the rapid development of an early prototype for M12, so that modelling, implementation and scenario refinement will continue. Small adjustments will be made to the approach where necessary. However, the presented approach to MyUI context management is simple and therefore flexible which make further adaptations easy to apply.

References

[OpenAAL09] Peter Wolf, Andreas Schmidt, Michael Klein: Applying Semantic Technologies for Context-Aware AAL Services: What we can learn from SOPRANO. Workshop on Applications of Semantic Technologies 09, Informatik 2009, Lecture Notes in Informatics vol. , GI, 2009

[SchmidtPhD09] Andreas Schmidt. Situationsbewusste Informationsdienste für das arbeitsbegleitende Lernen. Fakultät für Informatik, Universität Karlsruhe, PhD Thesis. 2009

[SchmidtContext] Andreas Schmidt. Ontology-based User Context Management: The Challenges of Dynamics and Imperfection. In Robert Meersman and Zahir Tahiri, editors, On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE. Part I., volume 4275 of Lecture Notes in Computer Science, pages 995–1011. Springer, 2006.

[StuderOntology] Steffen Staab and Rudi Studer. 2009. Handbook on Ontologies (2nd ed.). Springer Publishing Company, Incorporated.

[KresserReasoning] Sebastian Rollwage Kresser, Michael Klein and Peter Wolf. Collaborating context reasoners as basis for affordable AAL Systems. In: 4rd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI09), 2009.

[WolfSoprano] Peter Wolf and Andreas Schmidt and Michael Klein, Applying Semantic Technologies for Context-Aware AAL Services: What we can learn from SOPRANO; Workshop on Applications of Semantic Technologies 09, Informatik 2009

[WolfOpenAAL] Peter Wolf, Andreas Schmidt, Javier Parada Otte, Michael Klein, Sebastian Rollwage, Birgitta König-Ries, Torsten Dettborn, Aygul Gabdulkhakova openAAL - the open source middleware for ambient-assisted living (AAL) In: AALIANCE conference, Malaga, Spain, March 11-12, 2010, 2010

[DeyContext] Anind Dey, Gergory Abowd; Toward a better understanding of context and context-awareness; In HUC ’99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing

[UPsurvey] Sergey Sosnovsky, Darina Dicheva; Ontological technologies for user modeling; *Int. J. Metadata, Semantics and Ontologies, Vol. 5, No. 1, 2010*

[WinogradContext] Terry Winograd; Architectures for Context; HUMAN-COMPUTER INTERACTION, 2001, Volume 16, pp. 401–419

[HayesRothBB] Hayes-Roth, B.; A blackboard architecture for control; Artificial Intelligence, 1985, 26, 251-321.

[BoninoDogOnt] Dario Bonino und Fulvio Corno; DogOnt - Ontology Modeling for Intelligent Domotic Environments; The Semantic Web - ISWC 2008 ; Lecture Notes in Computer Science, 2008, Volume 5318/2008

[Totterdell1990] Totterdell, P.; Rautenbach, E. (1990): Adaptation as a Problem of Design, in: Browne, D.; Totterdell, P.

[Oppermann1094] Oppermann (1994) Oppermann, R. (1994): Adaptively supported daptability. International Journal

[Jameson2001] Jameson, A. (2001): Systems that adapt to their Users. Tutorial presented at IJCAI 2001. Saarbrücken: German Research Center for Artificial Intelligence (DFKI)

[Kobsa2004] Kobsa, A. (2004): Adaptive Methoden – Benutzermodellierung. Grundlagen der Information und Dokumentation (5th completely revised edition), München, K. G. Saur, S.299-302

[HeckermannGumo] Gumo – The General User Model Ontology by: Dominik Heckmann, Tim Schwartz, Boris Brandherm, Michael Schmitz, Margeritta von Wilamowitz-Moellendorff; User Modeling 2005 (2005), pp. 428-432

Appendix A – Email Client Scenario Stories

Story

1: HOME

Arthur wants to use his new Net TV to see if his daughter has sent him an email. He starts the Net TV by pressing the „on"-Button on his remote of which he knows the position.
Arthur selects the email service by choosing the relevant button on the screen by pressing the according number on the remote.

2: INBOX

The screen shows 3 Messages with the name of the sender and the subject of the mail in big letters but Arthur cannot read both of the texts.
He tries to get nearer to the Net TV (leans forward) to see if he might be able to read the text from a lesser distance.
The Net TV recognizes his movement and increases the font size.

Still, Arthur cannot read the texts so he tries to decrease the distance again.
The NetTV sees that it does not help to increase the font size any further and switches to pictures of the sender and a button to open the mail.

Artur recognizes one of the pictures as his daughter and presses the according button on the remote.

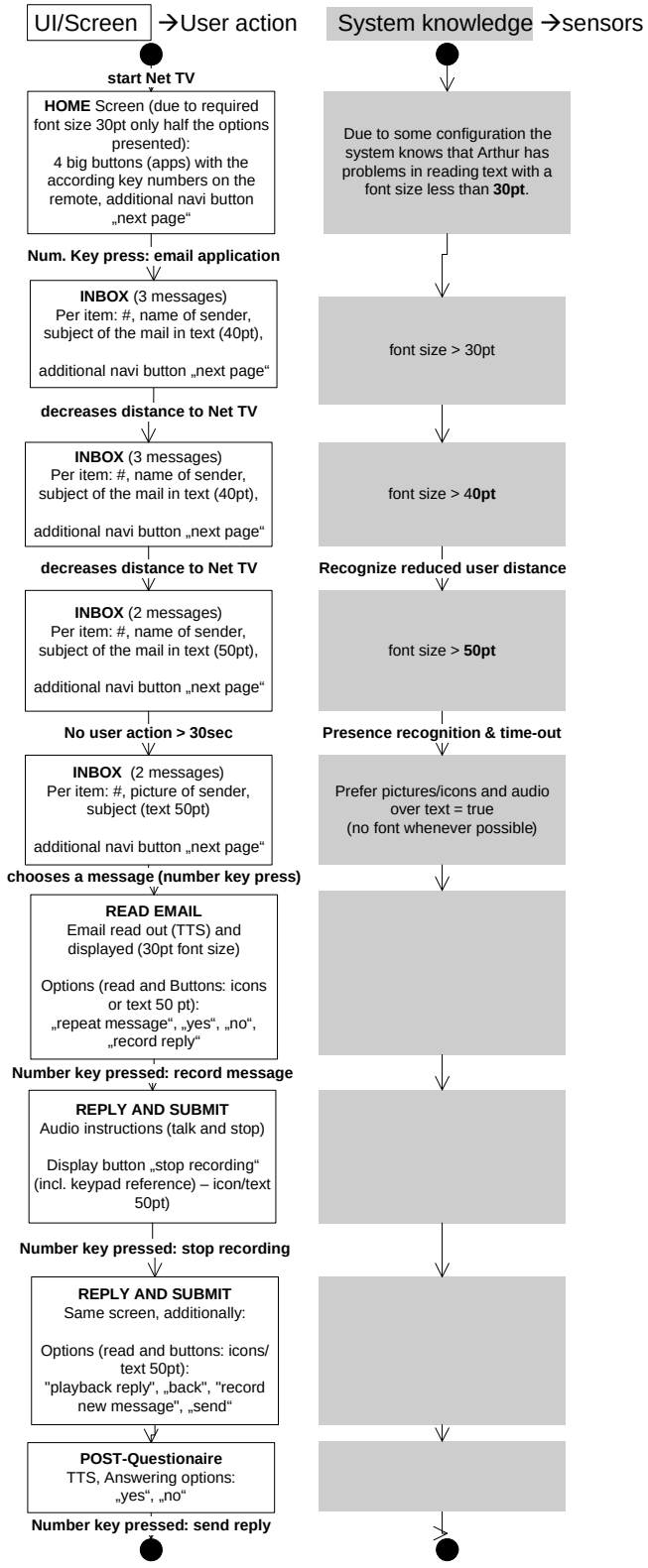
3: READ EMAIL

The Net TV switches to the message screen where the text as well as a button to read the message and different reply buttons are shown.
The message is read by the system (Text2Speach) automatically. After the system has read him his daughters message he also describes the options available to Arthur (audio menu).

4: REPLY & SUBMIT

His daughter asks him if a pickup for grocery shopping next Monday, 4 pm is okay for him, but since he has visitors at the time, he presses the button "record message" to record a reply to his daughter. The Net TV tells him that he can start to record after the „beep" and also tells him how to stop the recording when he is finished. Arthur waits for the „beep" and starts to speak his message.

After the email is sent, a written dialogue pops up to ask Arthur if the changes made to the interface while using the Net TV were acceptable to him.



Appendix B – Glossary of terms in MyUI

1. User model

The knowledge about the MyUI user and relevant context conditions is represented by an ontology of user interaction capabilities and constraints. As the main purpose of the MyUI user model is serving as a basis for user interface adaptations, perceptual, cognitive and motor user characteristics are deliberately defined in terms of interaction capabilities and constraints rather than on the level of medical diagnoses.

2. Classification of user models

According to [Dieterich et al., 1993][9] there are three major criteria to distinguish user models: granularity, temporal extent and representation.

In MyUI, a pragmatic, function-based, common user model for all users is applied. Within the MyUI user model, user and context characteristics which are relevant in order to adapt the user interface to users' needs and preferences are stored and updated. As the MyUI project focuses primarily on the target user groups of older people and stroke survivors, the current version of the user model is directed towards these (very heterogeneous) user groups. Hence, the MyUI user model meets the criteria of granularity (by the common user model for all users – also named canonical) and temporal extent (by storing the user characteristics beyond the session's duration – the so-called long-term user model approach).

A further classification dimension can be the purpose of user modelling. Possible purposes include:
* simulation (e.g. for illustration and evaluation purposes)
* diagnosis
* personalisation (e.g. user interface adaptation, recommender systems, personalised marketing and sales)

MyUI concentrates on individual user interface adaptation, basic simulation approaches are covered as a secondary target.

3. Application domains covered

MyUI application fields are:

- interactive TV platform services
- healthy exercise
- social communication
- entertainment
- accessibility support for developers and designers
- support for caregivers to create customised exercises

4. User Profile

According to ISO 9241-129 (Guidance on software individualization), paragraph 3.6, a user profile is defined as a set of attributes used by the system that are unique to a specific user/user group [ISO 9241-151:2008, 3.19].

In MyUI, individual user profiles are derived from the general user model and initial values are assigned to all attributes provided by the user model. User profile attributes are continuously updated over time based on the analysis of new sensor data and the individual user's interaction behaviour.

[9] [Dieterich et al., 1993] H. Dieterich, U. Malinowski, T. Kühme, M. Schneider-Hufschmidt: State of the Art in Adaptive User Interfaces, Adaptive User Interfaces: Principles and Practice, Elsevier Science Publishers, 1993, pp. 13-48.

5. Virtual User
A virtual reality-based representation of an instance of the user model (i.e. a virtual user profile).

6. Virtual Human
A digital human model that may be used to simulate the behaviour and actions of a physical user. However, simulation is not the main purpose of the MyUI project.

7. Simulation
In MyUI, simulation is understood as a time-based sequence of user interface events that mimics the behaviour and interaction patterns, actions and sensor readings of a real-life situation for the purpose of evaluating interface adaptation techniques.

8. Model Validation (Metrics and Process)
Model validation in MyUI is primarily done with regard to the self-adaptive ICT applications provided by the MyUI system. This is explicitly covered by enabling the user to access and modify its user profile at any time as needed. Additionally, the foreseen feedback loops to the user in case of adaptation of the user interface is also considered as an implicit way of model validation during user interaction with a MyUI application. Thereto different options of adaptation could be offered to the user or his confirmation could be asked for.
A further indirect way of user model validation can also be included in some of the planned user studies by collecting established usability and user acceptance measures during the use of MyUI applications.

9. Interface Adaptation
A recurrent, self-controlled process of changing the user interface design at runtime based on updates in the user profile. This aims at leading to a stepwise improvement of the individual user interface design according to the user's needs and preferences.

10. Profile Adaptation
According to [Dieterich et al., 1993][10] incremental updates of the user profile can be either done by the user or by the system. In MyUI, the refinement of the user profile is controlled by the system. Each time when new findings are gained from processed sensor input or the current user interaction behaviour, the MyUI user profile is refined. In this way, the accuracy of a specific user profile is aimed to be continuously improved at runtime.

Note: MyUI user profiles result from complex sensing/recognition and interpretation algorithms which are error prone. Therefore, the current values of a MyUI user profile are inherently probabilistic and have a considerable level of uncertainty. This contrasts to other (more diagnosis-oriented) user profiling approaches where user profiles can be determined with much higher levels of reliability.

11. (User Interface) Design Pattern
Design patterns in MyUI cover all user interface design aspects that need to be adapted in order to suit the different MyUI users' requirements. Main components of a pattern include the problem description, the context and the solution of a specific recurring design problem as well as references and related patterns. Four different kinds of patterns are distinguished in MyUI:

1. Generic design patterns:
   They translate user characteristics into user interface features which are set by global variables. These features are stored in the user interface profile.

[10] [Dieterich et al., 1993] H. Dieterich, U. Malinowski, T. Kühme, M. Schneider-Hufschmidt: State of the Art in Adaptive User Interfaces, Adaptive User Interfaces: Principles and Practice, Elsevier Science Publishers, 1993, pp. 13-48.

2. Interaction design patterns:
   They describe variants of controls or elements which are related to a specific interaction situation and user interface profile.
3. Common patterns:
   Basic features of MyUI user interfaces which do not change during the interface adaptation process are specified in these patterns.
4. Transition patterns:
   These describe how to switch between single patterns of a bundle of related generic or interaction design patterns.

12. User Interaction

User interaction in MyUI is the direct way a user communicates with the applications of the MyUI system by means of one of the supported input devices. Besides the input from input devices, the analysis of more "implicit" user interaction is an essential element to determine changes in the user profile. "Implicit human-computer interaction" refers to human behaviour and actions that are not explicitly directed towards a computer interface and are recognized and interpreted as a trigger of system reactions. Examples from the MyUI project include gestures (e.g. lean-to-screen), head position and gaze direction, time-out, etc.

13. Disablility, capabilities and functional limitations

A note on this contribution:
'Impairment' has been added as an additional glossary term, as this is used repeatedly throughout MyUI documents and is used more widely in conjunction with disability to give nuance to the different facets of disabled experience. As there is no universally agreed theory of disability, the following glossary entries draw upon several different epistemologies (the bio-psycho-social model and the capability approach) to pragmatically account for those facets of disability that are most relevant to the project. These terms aim to use an ability, rather than deficit, model. For this reason, Functioning Limitation is re-defined as Functioning. An additional glossary entry for Functioning Limitation could be added, but we think this is self explanatory – and Impairment may represent a more accessible term for the same meaning. References are added for information.

Glossary entries:

Capability identifies the freedom of the individual to achieve valuable functionings in a given circumstance. In this way, capability is not the presence of a physical or a mental ability; instead, it is understood as a practical opportunity.
(Drawing on: Sen, A. (1999: 74), Development as Freedom. Oxford: Oxford University Press and Mitra, S. (2006:54) The Capability Approach and Disability, Journal of Disability Policy Studies 2006 16: 236)

Disability is a disadvantage or marginalisation that results from the interaction between people with impairments and the attitudinal and environmental barriers that obstruct their equal participation in society. Disability is a social construct, not an attribute of an individual. Society and context determine who experiences disability.
(Abridged and expanded from UN Convention on the Rights of Persons with Disabilities).

Functioning identifies the cognitive and physical actions available to an individual.
(Based upon WHO (2002) Towards A Common Language For Functioning, Disability and Health International Classification of Functioning, Disability and Health)

Impairment identifies a physical, sensory or cognitive limit that reduces functioning.
(adapted from WHO, 2002)