



## MyUI: Mainstreaming Accessibility through Synergistic User Modelling and Adaptability

FP7-ICT-2009-4-248606

### Prototype for user context management infrastructure and user modelling

Public Document

<b>Deliverable number</b>	D1.2	<b>Date of delivery</b>	12-2011
<b>Status</b>	Final	<b>Type</b>	Prototype
<b>Workpackage</b>	WP1 - User and Context Modelling		
<b>Authors</b>	UC3M (José Alberto Hernández, David Larrabeiti), FZI (Oliver Strnad, Andreas Schmidt)		
<b>Keywords</b>	Context Manager; Ontology; User Profile; XML-RPC; C#		
<b>Abstract</b>	This deliverable reviews the basic concepts and implementation of the Context Manager for the MyUI system architecture. The Context Manager is designed to capture the user's profile and context and provide this information to the MyUI services such that, the User Interface offered to the end-user can be individualised to a particular user, taking into account his/her perceptual, cognitive and motor impairments. This deliverable is intended to serve as an update of the previous D1.1, showing the details and the refined Context Manager of the MyUI system.		

## Table of Contents

<b>1. INTRODUCTION AND SCOPE.....</b>	<b>3</b>
<b>2. BRIEF REVIEW OF THE CONTEXT MANAGER AS DESCRIBED IN D1.1 .....</b>	<b>4</b>
2.1 USER PROFILE .....	4
2.2 SENSOR DATA .....	5
<b>3. CONTEXT MANAGER - IMPLEMENTATION AND FUNCTIONALITY .....</b>	<b>6</b>
3.1 CONTEXT MANAGER UPDATE .....	6
3.1.1 Architectural changes.....	6
3.1.2 Graphical User Interface .....	7
3.1.3 Development Support .....	11
3.1.4 Finalization of the Context Manager API .....	11
3.1.5 More Flexible Data Model .....	15
3.2 SENSOR ONTOLOGY UPDATE.....	15
3.2.1 Application-Specific Sensors .....	15
3.2.2 Timeout Sensor .....	17
3.2.3 Attention Sensor .....	17
3.2.4 TmtGameSensor .....	17
3.2.5 CorsiTestSensor.....	18
3.2.6 Current State of the Sensor Ontology.....	18
<b>4. FINALIZATION OF THE USER PROFILE.....</b>	<b>18</b>
4.1 FINAL VERSION.....	18
4.1.1 User Profile Variable: Successful Interactions .....	21
4.1.2 User Profile Variable: State transitions.....	21
4.1.3 User Profile Variable: Experience .....	21
4.2 DEVELOPMENT OF AN INSTANCE OF THE USER PROFILE.....	21
4.2.1 User Profile Drift .....	22
4.3 MANUAL ADAPTATION OF THE USER PROFILE.....	24
<b>5. THE ROLE OF GAMES FOR USER PROFILING .....</b>	<b>26</b>
<b>6. SUMMARY AND DISCUSSION .....</b>	<b>29</b>
<b>REFERENCES.....</b>	<b>30</b>

## 1. Introduction and Scope

As specified in the Description of Work (DoW) page 91, this deliverable attempts to provide a:

*“software prototype of user and context modeller, including a description of the refined context ontology and user modelling concept.”*

As such, this deliverable refines the MyUI’s context management system described in D1.1. Recalling from D1.1 the MyUI system must provide a Context Management system that collects and aggregates contextual information about the user and his/her environment (via sensors), and stores that information in a user profile . This user profile provides the base for the adaptation of the User Interface (UI) to the particular context of the user, thus providing individualised services.

The previous Deliverable D1.1 described a preliminary version of the Context Management system (or Context Manager) as originally conceived and implemented by Month 12. The Context Manager has been further refined, both in concept and implementation. This deliverable summarises the main changes and updates of the Context Manager, and its involvement in the MyUI project. Section 2 briefly reviews the Context Manager concept and the new updates introduced. Section 3 further describes the implementation of the Context Manager. It also provides concepts for sensors introduced after deliverable D1.1 and examples about how to use the updated version of the Context Manager. Section 4 proposes the final selection of user profile variables discusses newly added user profile variables and illustrates the development of an instance of a user profile for a user over time. At the end of the section the user profile drift – an inherent problem of the user profiling system used in MyUI is discussed with solutions showing the problem can be handled. Also the possibility and corresponding side-effects of manual updates of the user profile will be shown. Section 5 includes a brief description of the role of the Context Manager in Task 4.5, about the use of games for user profiling. Finally, Section 6 concludes this deliverable.

## 2. Brief Review of the Context Manager as Described in D1.1

In D1.1 the general concepts of the Context Manager have been described. This chapter recalls some of these concepts.

The Context Management system captures the situation of a user. More concretely:

*“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 [1]).*

For MyUI the Context Manager manages user profile data and context data in the form of sensor information. Its main purpose is capturing and aggregating sensor events as well as the adaptation of the user profile based on this information.

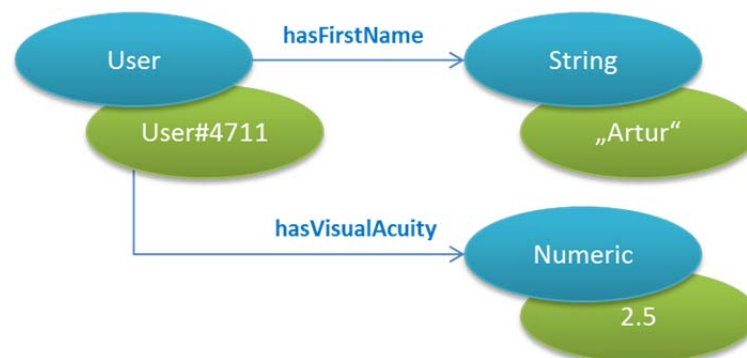
The managed data is captured in a structured way and defined in ontology schemas. For this purpose we presented the User Profile Ontology and the Sensor Ontology in D1.1. Both will be refined later in the document.

The Context Manager was implemented on top of the open source middleware OpenAAL developed within the European-founded project SOPRANO. Since the software of the MyUI system consists of multiple distinct parts where each part has a connection to the Context Manager it was decided to provide an API for the Context Manager. For this task a XML-RPC interface has been implemented to support a platform- and programming-language-independent communication between all components. The interface of the Context Manager is described in D1.1 and D4.2.

### 2.1 User Profile

The User Profile is a collection of information about the user of the MyUI system. It consists of static data like email address, first name, last name, etc. and information particularly important for the user interface customization. To achieve this, the MyUI user profile focuses on possible disabilities and impairments related to the usage of user interfaces.

As described in Figure 1 there are multiple levels of abstraction involved. On the ontological level we have classes like “User”, properties like “hasName” and datatypes like “String”. This level is called the user profile. The next level of abstraction is the concrete instantiation of this user profile for a given user. Each triple of an instance of the class “User”, a property and an instantiation of e.g. “String” is called a user profile variable value.



**Figure 1: Example showing a part of the user profile. The blue circles present the schema level, the arrows indicate the properties and the green circles are the instances of the corresponding classes that form a concrete instance of the user profile.**

The methodology used to create the user profile ontology and a detailed description of the RDF-based data model can be found in D1.1.

## 2.2 Sensor Data

Since one of the goals of MyUI is to provide a suitable adaptation of the user interface to the user it is important to also adopt the user profile. This means that the aforementioned disabilities and impairments, or at least their symptoms in terms of UI utilization, need to be measured constantly. Based on these measurements the user profile is adopted.

In order to integrate the ability of measuring the properties of a given user the Context Manager captures and aggregates structured sensor data modelled in the sensor data ontology. Two different types of sensors are distinguished at this point: physical sensors and virtual sensors. While physical sensors are real devices such as a webcam or a RFID-reader, virtual sensors are mainly implemented in software. The Context Manager captures the measurements of the different sensors in form of a so called sensor event which is basically a triple of sensor identifier, the measured property and a value for that measured property. A sensor event can be e.g. that a webcam sensor measured a distance of 2.5 meters between the display and the user.

In order to enable the Context Manager to automatically derive useful information from the sensors, meta-information about each sensor is needed. The meta-data contains statements like which user carries a given RFID-transponder and to which installation the RFID-reader that detected the transponder is connected.

### 3. Context Manager - Implementation and Functionality

The main concept of the Context Manager is basically the same as originally conceived in the project and reported in D1.1 (and further reviewed in the previous section). However, the original ontology for the sensor data has been extended with new concepts as ideas came up for further measurements of the user, like measuring the attention of the user by tracing the direction he is looking at. The sensor ontology has also been extended with new attributes for the data originating from the games described later in this deliverable.

Furthermore a new feature was included into the adaptation engine, which allows the user of the MyUI application better control over the adaptation process by giving him the ability to reject an adaptation of the user interface that took place.

The rationale behind this is that measurements of the user's disabilities can be wrong due to outer factors influencing the system. For example the user profile variable for attention could indicate that the user is inattentive. But the measurements just showed the user as inattentive because he was looking at someone entering the room. In this case the adaptation of the user interface could be inappropriate for the user and the user would have the ability to tell the system that he was satisfied with the adaptation he had before the change. Support for this mechanics has also been included in the Context Manager by extending the sensor ontology.

#### 3.1 Context Manager Update

As of D1.1 the Context Manager was a monolithic application on top of OpenAAL running on a dedicated server at FZI in Karlsruhe. The drawbacks of such a solution were that every partner needed access to the internet in order to develop or test the MyUI framework. Furthermore the internet connection used to transfer the user profile data and sensor events between the adaptation engine and the Context Manager introduced heavy delays to the adaptation of the user interface, which was unacceptable for the adaptation tasks the MyUI framework carries out. During discussions inside the project it was found that there was a demand for installing local instances of the Context Manager in order to support debugging and development of the other components. There were also concerns regarding data security associated with the centralized storage of context information, which will also be targeted by the following features.

To address these issues it has been decided to create a new implementation of the Context Manager including the following points:

- Replace the centralized architecture of the system by an architecture where every installation of the MyUI system has its own instance of the Context Manager running.
- Graphical User Interface to configure sensors, create users and change the user profile
- Development support by giving detailed information about what is happening inside the Context Manager and by providing access to the underlying database
- Finalize the API of the Context Manager used by other components of the MyUI system for the communication with the Context Manager
- Engineer a more flexible data model

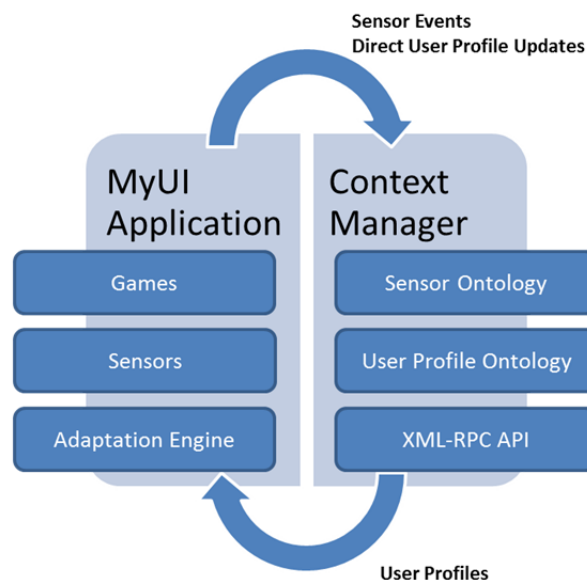
##### 3.1.1 Architectural changes

As described above the centralized architecture implemented in MyUI had certain drawbacks regarding usability, configuration and processing speed. There was only one instance of the Context Manager running for all instances of the MyUI framework. While this approach worked for the development of the first prototype it showed to limit the further development of the framework.

Also, the centralized approach made debugging and error tracing difficult, since a secure remote desktop connection to the server running the Context Manager would have been needed to read the error logs. Even more difficult to handle was a situation where simultaneous access to development APIs from multiple partners to the Context Manager appeared. This resulted in simultaneous and conflicting updates of the user profile for a given user.

To overcome these limitations a new Context Manager has been implemented in C# to improve its performance and benefit from the rapid application development approach of Visual Studio for C# when it comes to graphical user interfaces. It has been decided that the software will be implemented as a stand-alone application each project partner can use on their own systems without relying on a slow socket connection over the public internet like the old version of the Context Manager did. Furthermore the database has been replaced by SQLite which is an embeddable relational database supporting SQL and which is used in multiple large-scale software projects like Photoshop Lightroom by Adobe. The data format of SQLite databases is open and can therefore be read and modified by a broad range of utilities.

By providing the ability to replace the database of the Context Manager it is now possible to have different databases for different studies and archive the results of these studies, as well as providing pre-configured databases to project partners.



**Figure 2: Simplified overview of a complete installation of the MyUI framework.**

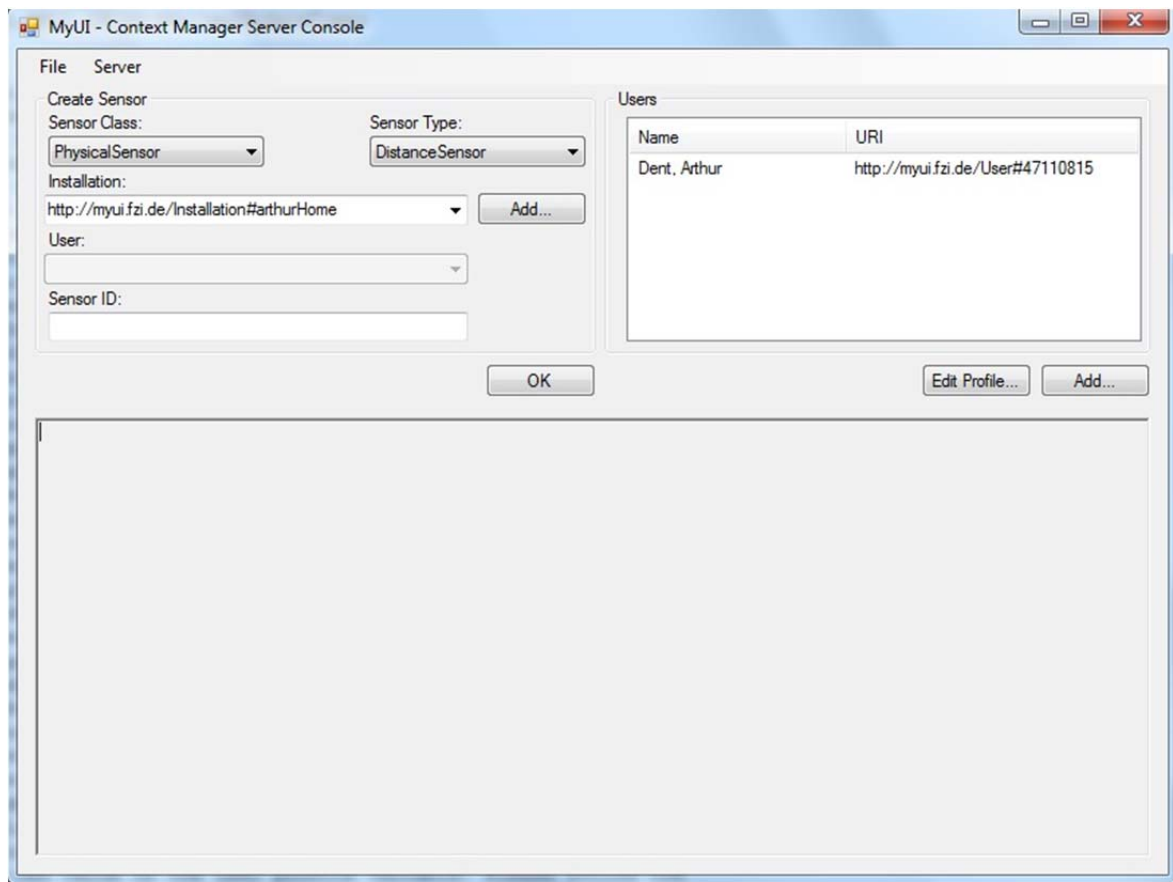
### 3.1.2 Graphical User Interface

By having one instance of the Context Manager for every installation of the MyUI framework the control over the configuration of these installations was also transferred to the project partners in order to ease the creation of demo-installations, the integration of new sensors or to test adaptations of the MyUI User Interface a graphical user interface has been created for the Context Manager.

Partners running an own instance of the Context Manager can now create new databases or open already existing ones. Furthermore it is now possible to edit the database with the graphical user interface. Maintenance operations like adding a user, creating a new sensor or editing the user profile of a user are now easy to accomplish.

Figure 3 shows the main menu of the Context Manager GUI. The File menu allows creating a new database or opening an already existing one (for example, the one provided is “myui\_example.db”).

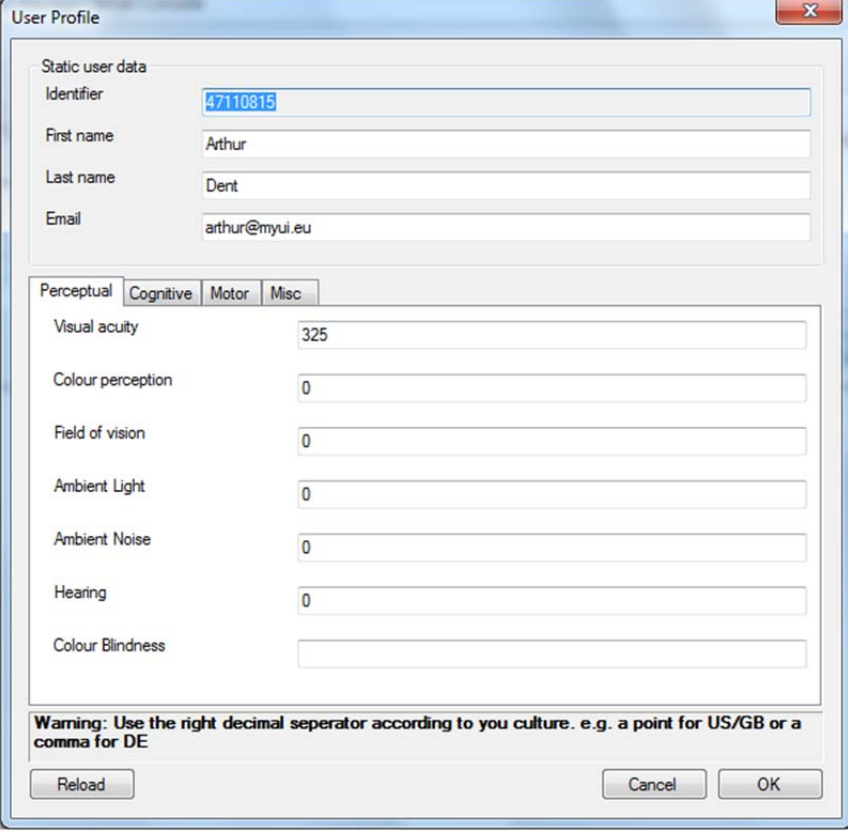
After this, the Server menu allows to start the Context Manager server. Once the database is opened and the server is started, we can see from the Users list a number of users loaded in the database. In this example, there is only one user in the database: Arthur Dent, identified by his id, namely 47110815. You can edit his profile by clicking the “Edit Profile” button.



**Figure 3: Main Menu of the Context Manager Server Console**

After clicking the Edit Profile button, the menu of Figure 4 is shown. This figure allows modifying the user’s id, name and email address. Additionally, there is a list of Perceptual attributes and their current values, including: Visual Acuity, Colour perception, Field of vision, etc. The Menu also shows other Cognitive related properties, Motor properties and Misc properties, along with their values (see Figure 5, Figure 6 and Figure 7).





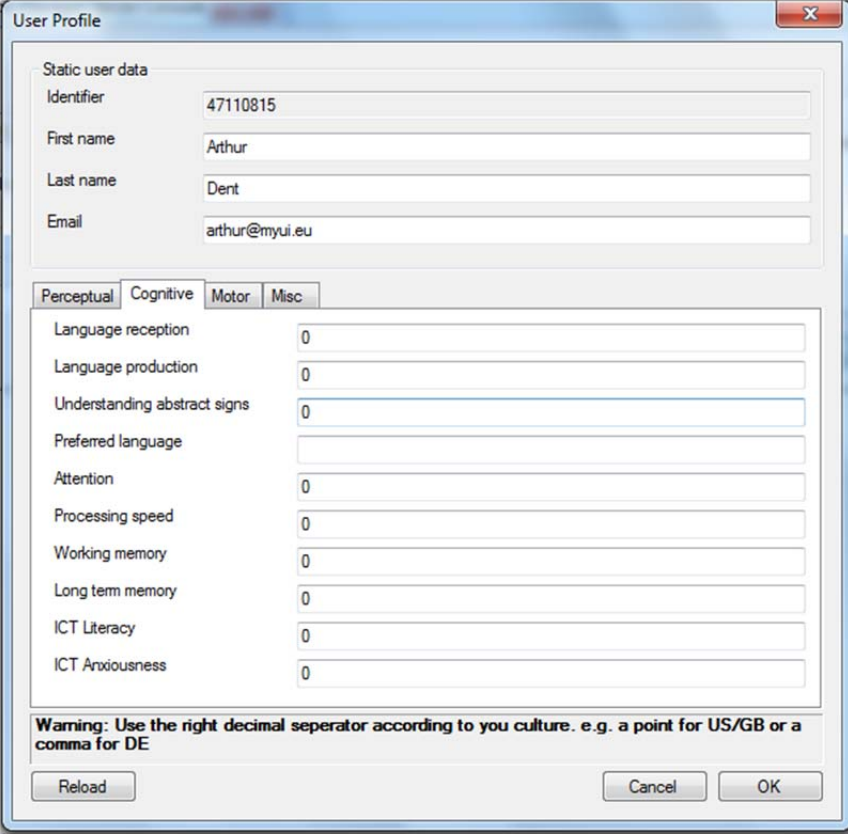
The 'User Profile' dialog box displays the 'Perceptual' tab. It contains a 'Static user data' section with fields for Identifier (47110815), First name (Arthur), Last name (Dent), and Email (arthur@myui.eu). Below this is a table of perceptual properties with values entered in the input fields.

Property	Value
Visual acuity	325
Colour perception	0
Field of vision	0
Ambient Light	0
Ambient Noise	0
Hearing	0
Colour Blindness	

Warning: Use the right decimal separator according to you culture. e.g. a point for US/GB or a comma for DE

Buttons: Reload, Cancel, OK

Figure 4: User Profile Perceptual Properties



The 'User Profile' dialog box displays the 'Cognitive' tab. It contains the same 'Static user data' section as Figure 4. Below this is a table of cognitive properties with values entered in the input fields.

Property	Value
Language reception	0
Language production	0
Understanding abstract signs	0
Preferred language	
Attention	0
Processing speed	0
Working memory	0
Long term memory	0
ICT Literacy	0
ICT Anxiousness	0

Warning: Use the right decimal separator according to you culture. e.g. a point for US/GB or a comma for DE

Buttons: Reload, Cancel, OK

Figure 5: User Profile Cognitive Properties

User Profile

Static user data

Identifier

47110815

First name

Arthur

Last name

Dent

Email

arthur@myui.eu

Perceptual

Cognitive

Motor

Misc

Speech articulation

0

Finger precision

0

Hand precision

0

Arm precision

0

Contact grip

0

Pinch grip

0

Clench grip

0

Warning: Use the right decimal seperator according to you culture. e.g. a point for US/GB or a comma for DE

Reload

Cancel

OK

Figure 6: User Profile Motor Properties

User Profile

Static user data

Identifier

47110815

First name

Arthur

Last name

Dent

Email

arthur@myui.eu

Perceptual

Cognitive

Motor

Misc

Hand-eye coordination

0

Warning: Use the right decimal seperator according to you culture. e.g. a point for US/GB or a comma for DE

Reload

Cancel

OK

Figure 7: User Profile Misc Properties

### 3.1.3 Development Support

The new graphical user interface contains a part dedicated to provide logging information. The user of the Context Manager is now able to see in real-time which statements are stored into the database and what inference steps the Context Manager performs internally.

Furthermore the SQLite-format allows opening the database files with 3<sup>rd</sup> party software like SQLite Manager (see Figure 8). This enabled developers to get all the raw information inside the MyUI Context Manager and make tweaks for debugging purposes.

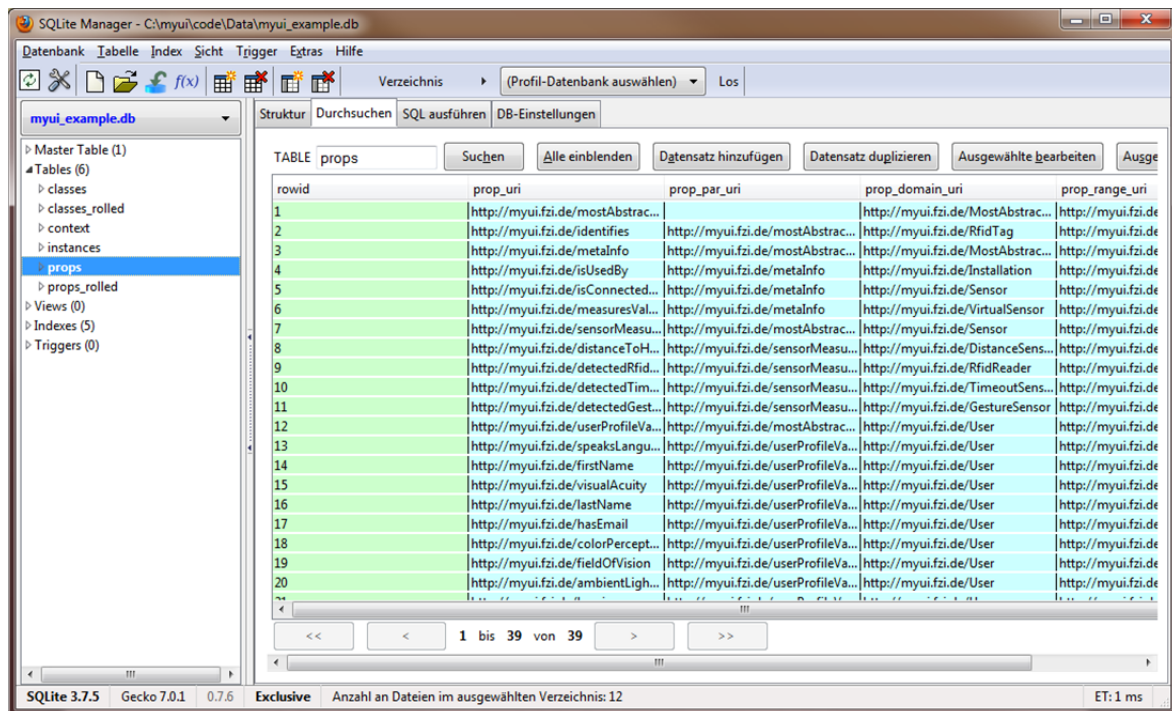


Figure 8: Opening the database with SQLite Manager (German Screenshot)

### 3.1.4 Finalization of the Context Manager API

The Context Manager provides an interface to the other parts of the MyUI framework, offering them the possibility to query, modify or delete information from the Context Manager. Since MyUI consists of multiple applications implemented in different programming languages (e.g. C#, PHP) an easy to use cross-platform standard for remote procedure calls had to be chosen.

As a solution the XML-RPC protocol, which is the precursor of the SOAP protocol, fits these requirements. XML-RPC is a text-based protocol transported on top of the HTTP protocol. In contrast to SOAP the protocol is very simple and should therefore be easy to understand by developers. There are implementations for most of the commercially used programming languages which makes it portable across different platforms. The reader is referred to Deliverable D4.2 for a brief review of XML-RPC.

The following table presents the API for the Context Manager which is exposed via XML-RPC. For each method its name, the expected parameters, a description of the return value as well as a short example is given. All examples provided are written in Java-like pseudo code.

<b>Method</b>	<b>myui.PublishSensorEvent</b>	
<b>Description</b>	<p>The method PublishSensorEvent can be called by a client to post a detected sensor event to the Context Manager of MyUI in the form of a context triple. Currently the systems is capable to process sensor events from the following sensors:</p> <ul style="list-style-type: none"> <li>• Distance Sensor measuring the distance between the head of the user and the display device</li> <li>• Gesture Sensor detecting lean-forward and lean-backward gestures of the user</li> <li>• RFID reader detecting whether a RFID transponder is in range or not</li> <li>• Attention sensor detecting whether the user is looking at the display of the MyUI system</li> <li>• Timeout sensor detecting whether the user executes no actions inside the MyUI application for a given time</li> <li>• A sensor capturing the results of the Trail-Making-Game</li> <li>• A sensor capturing the results of the Corsi-Test-Game</li> </ul> <p>Each sensor can detect different properties of its environment. Therefore the following properties were defined for the existing set of sensors:</p> <ul style="list-style-type: none"> <li>• distanceToHead</li> <li>• detectedGesture</li> <li>• detectedRfidTag</li> <li>• detectedTimeout</li> <li>• detectedCorsiScore</li> <li>• detectedCorsiDetailScore</li> <li>• detectedTmtTime</li> <li>• detectedTmtMisclicks</li> </ul>	
<b>Parameters</b>	1 <sup>st</sup> parameter	The first parameter given to this method is of type string and holds the identifier of the sensor that emitted the sensor event.
	2 <sup>nd</sup> parameter	The second parameter is of type string and holds the property that has been measured by the sensor.
	3 <sup>rd</sup> parameter	The third parameter is also of type string and holds the value of the measured property.
<b>Return value</b>	This function returns a Boolean that indicates whether the call succeeded and the sensor event has been stored inside the Context Manager.	
<b>Example</b>	<pre>// Publish that the sensor with the identifier Distance01 // detected a distance to the head of the user of 25.0 cm Myui.PublishSensorEvent("Distance01",     "distanceToHead", "25.0");  // Publish that the sensor Camera01 detected a lean-forward // gesture Myui.PublishSensorEvent("Camera01",     "detectedGesture", "Forward");</pre>	

<b>Method</b>	<b>myui.GetUserProfile</b>	
<b>Description</b>	The method GetUserProfile allows the client to retrieve a complete user profile of a given user.	
<b>Parameters</b>	1 <sup>st</sup> parameter	The first parameter given to this method is of type string and holds the identifier of the user whose complete user profile shall be retrieved.
<b>Return value</b>	This method returns a list of Strings where each String contains all data about one specific user profile variable. The data is transmitted as XML. If no user profile for the given user identifier was found an empty list is returned.	
<b>Example</b>	<pre>// Retrieve the user profile of "Artur" Myui.GetUserProfile("Artur")</pre> <p>The following two entries of the user profile show how each string in the list is constructed:</p> <pre>&lt;varName&gt;LastName&lt;/varName&gt; &lt;varValue&gt;Dent&lt;/varValue&gt; &lt;varUpdateTime&gt;Thu Nov 17 16:52:36 CET 2011&lt;/varUpdateTime&gt;  &lt;varName&gt;FirstName&lt;/varName&gt; &lt;varValue&gt;Artur&lt;/varValue&gt; &lt;varUpdateTime&gt;Thu Nov 17 17:22:45 CET 2011&lt;/varUpdateTime&gt;</pre>	

Method	myui.UpdateUserProfile	
Description	The method UpdateUserProfile allows the client to directly set the value of a user profile variable for a given user.	
Parameters	1 <sup>st</sup> parameter	The first parameter given to this method is of type string and holds the identifier of the user whose complete user profile shall be retrieved.
	2 <sup>nd</sup> parameter	The second parameter is of type string and contains the name of the user profile variable (eg. LastName).
	3 <sup>rd</sup> parameter	The third parameter is of type string and contains the new value for the user profile variable given in parameter two.
Possible Values	Possible values for the second parameter are: FirstName (String) LastName (String) Email (String) VisualAcuity (Numeric, [0,4]) FieldOfVision (Numeric, [0,4]) AmbientLight (Numeric, [0,4]) Hearing (Numeric, [0,4]) AmbientNoise (Numeric, [0,4]) LanguageReception (Numeric, [0,4]) LanguageProduction (Numeric, [0,4]) UnderstandingAbstractSigns (Numeric, [0,4]) Attention (Numeric, [0,4]) ProcessingSpeed (Numeric, [0,4]) WorkingMemory (Numeric, [0,4]) LongTermMemory (Numeric, [0,4]) SpeechArticulation (Numeric, [0,4]) FingerPrecision (Numeric, [0,4]) HandPrecision (Numeric, [0,4])	

	ArmPrecision (Numeric, [0,4]) ContactGrup (Numeric, [0,4]) PinchGrip (Numeric, [0,4]) ClenchGrip (Numeric, [0,4]) HandEyeCoordination (Numeric, [0,4]) ICTLiteracy (Numeric, [0,4])
<b>Return value</b>	The method returns a Boolean indicating whether the call succeeded.
<b>Example</b>	<pre>// Update the user profile variable "VisualActuicity" of // a user with the identifier "Artur" to a value of 4.0 Myui.UpdateUserProfile("Artur", "VisualAcuity", "4.0");</pre>

<b>Method</b>	<b>myui.GetUserVariable</b>	
<b>Description</b>	The method GetUserVariable allows the client to retrieve the value of a given user profile variable. This method was introduced for convenience of the developer to allow the retrieval of user profile variables without parsing the complete user profile.	
<b>Parameters</b>	1 <sup>st</sup> parameter	The first parameter given to this method is of type string and holds the identifier of the user whose user profile shall be retrieved.
	2 <sup>nd</sup> parameter	The second parameter is of type string and holds the name of the user profile variable that shall be retrieved.
<b>Return value</b>	This method returns a single String that contains the value of the user profile variable given in the second parameter.	
<b>Example</b>	<pre>// Retrieve the user profile variable "VisualAcuity" of "Artur" Myui.GetUserVariable("Artur", "VisualAcuity");</pre>	
	This call returns the string "4.0" if "Artur" has a visual acuity of "4".	

<b>Method</b>	<b>myui.GetUserVariableHistory</b>	
<b>Description</b>	The method GetUserVariableHistory allows the client to retrieve a list of triples that shows the development of a user profile variable for a given user.	
<b>Parameters</b>	1 <sup>st</sup> parameter	The first parameter given to this method is of type string and holds the identifier of the user whose user profile shall be retrieved.
	2 <sup>nd</sup> parameter	The second parameter is of type string and holds the name of the user profile variable that shall be retrieved.
<b>Return value</b>	<p>This method returns a list of strings where each string contains one user profile statement in the form of a triple consisting of the user identifier, the name of the user profile variable, the value of the user profile variable and the time it was set to this value.</p> <p>If a user profile variable was not yet explicitly set to a value for the given user a standard value is returned and the time of the last change of this user profile variable is the current timestamp.</p>	
<b>Example</b>	<pre>// Retrieve the user profile variable history of // "VisualAcuity" for the user "Artur" Myui.GetUserVariableHistory("Artur", "VisualAcuity");</pre>	
	<p>This call could return the following XML-snippet:</p> <pre>&lt;profileStatement&gt;   &lt;userId&gt;Artur&lt;/userId&gt;</pre>	

	<pre> &lt;userVariable&gt;VisualAcuity&lt;/userVariable&gt; &lt;value&gt;3.25&lt;/value&gt; &lt;timeSet&gt;02.08.2011 14:41:03&lt;/timeSet&gt; &lt;/profileStatement&gt; </pre>
--	---

### 3.1.5 More Flexible Data Model

In the old version of the Context Manager the ontology schema had to be pre-defined at design time of the Context Manager. This resulted in a lot of minor updates of the Context Manager component– one for each update of the ontology schemas. Since MyUI will be a framework providing other developers with the ability to develop own applications on top of MyUI it has been found that functionality to programmatically change the underlying ontology schema is needed.

As described in deliverable D1.1, the ontology schema is composed of classes and properties similar to RDF Schema. As part of the new Context Manager an API has been implemented to create new classes and properties during run-time. This allows developer tools or even context augmentation services which generate contextual information of a higher abstraction level from low level context events to programmatically alter the schema.

Although the addition of new concepts to the ontology schema is now possible the implementation doesn't support the removal of concepts from the schema. This behaviour is intended since the Context Manager is designed to never “forget” instantiations of these concepts (the captured information). Therefore removing a concept from the ontology schema would mean to lose all instances of this concept which would work against the rule of never forgetting information.

## 3.2 Sensor Ontology Update

This chapter provides information about the extensions of the sensor ontology integrated into the Context Manager since deliverable D1.1. Each section will present the concept of one newly added sensor and explains how the sensor data is acquired and how the data originating from this sensor will be used to update the user profile inside the Context Manager.

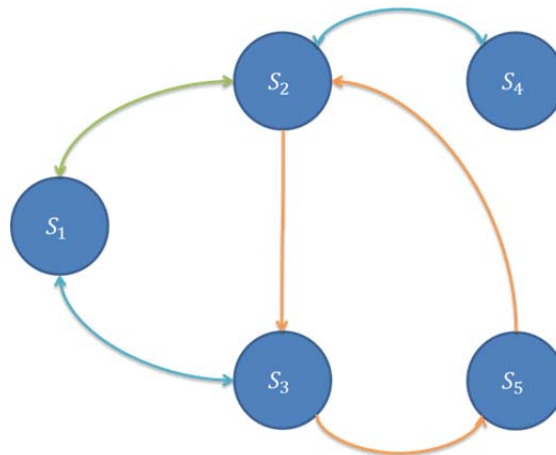
### 3.2.1 Application-Specific Sensors

This chapter describes the concepts of application-specific sensors. An application specific sensor is a virtual sensor which relies on information the application developer provides to the MyUI system. The main purpose of this type of sensors is to detect usage patterns which indicate that the user has problems using a given application. Since these problems vary between different applications – e.g. a successful interaction in an email-application can be something completely different than a successful interaction in an instant messaging application - the application developer has to provide these software-sensors with his application.

Each concept presented in this section comes with examples for an implementation in the MyUI email-application presented in D4.2 / R4.3.2.

#### 3.2.1.1 Circles

While navigating through an application the user could have problems and start navigating in circles. One possible scenario is where the user forgets where inside the structure of the application he currently is. Since MyUI applications are already organized in states and directional transitions between these states the detection of circular navigation fits well with the system.



**Figure 9: Example of a circular navigation inside an application. The user starts with state S1 and navigates to S2. From there he starts navigating from S3 to S5 and back to S2 because he cannot remember how to reach S1 again.**

For this task the system could analyze applications at deployment time for occurrences of circular transitions between the application states. Whenever the application is used during run-time the system has to check for occurrences of the sequences found during the analysis of the application.

### 3.2.1.2 Repeated Back

The repeated back can be seen as a special case of circular navigation. While the circular navigation detection takes circles introduced by direct links in the navigation of the application into account (see e.g. Figure 9:  $S_1, S_2, S_3, S_5, S_2, \dots$ ) the repeated back tracks the usage of the “Back”-button available in all applications of the MyUI framework.



**Figure 10: Example of a possible application with three states and two bidirectional transitions between states.**

While using the “Back”-button to quickly navigate through an application, e.g. to get back to the main menu, is not a behavior that should be tracked there are situations where a repeated usage of the “Back”-button could indicate a problem the user has with the structure of an application. Figure 10 shows a very simple application with three possible states. When entering the application the user is in state  $S_1$ . From there he can move to  $S_3$  via  $S_2$  or go back from  $S_3$  to  $S_2$  or from  $S_2$  to  $S_1$ . If the user starts a sequence like  $(S_1, S_2, S_1, S_2)$  this behavior could indicate one of the following:

- The user cannot remember which information  $S_1$  contained and needs to switch back between  $S_2$  and  $S_1$  to the task in  $S_2$ .
- The user is lost in the structure of the application either because he doesn’t understand it or because he cannot recall it.

### 3.2.1.3 Successful Interaction

This sensor detects whether the user achieved a pre-defined goal in the application with the user interface adaptation he is currently using. These goals are defined by the developer of an application and are regarded as successful interactions. A goal can be reached by a single action or by a sequence of actions the user has to perform.

If we take the email application as an example a successful interaction could be writing an email. This goal would consist of several steps the user would have to perform:

1. Select the email application from the main menu



2. Select “New” from the menu of the email application
3. Select the recipient for the mail
4. Enter a subject
5. Enter text for the body of the mail
6. Press the “Submit” button

After completing these steps the sensor would send a sensor event to the context manager. But what can the system infer about the user from this information?

The assumption is that the total number of successful interactions is very low for a bad adaptation and becomes higher with the quality of the adaptation rising over time. The number of successful interactions is therefore an indirect measurement of the adaptation quality.

However there are several points to take into account. The first point is that the system has to know the starting point of a sequence of interactions, which can be resolved by taking all actions since the last successful interaction into account. Also the number of successful interactions alone does not provide a reliable approximation of the adaptation quality. It is possible that the user reaches a pre-defined goal for the application by trial-and-error or via a large amount of steps which would mean that the adaptation is not well-suited for the user. On the other hand a low number of steps to reach a certain goal would indicate that the user knows what he is doing and is able to use the user interface without problems, which would mean that the adaptation is well-suited for the user at hand.

### 3.2.2 Timeout Sensor

The timeout sensor is a virtual sensor associated with a given end-user issuing an event whenever the end-user of the MyUI system is inactive for a certain period of time. It is currently realized as a JavaScript inside the MyUI application an end-user is going to see on his display device (e.g. the television).

The idea behind measuring a timeout in the application usage is that the user will hesitate if something is unclear to him respectively he cannot understand what is presented to him or if the user is inattentive. These are signs that the adaptation is not suitable for the user and he should be provided with another adaptation with e.g. less information displayed or higher font size.

In the current implementation the Context Manager adopts the user profile variables for visual acuity, language reception and attention if the timeout sensor issues a sensor event. With this event the Context Manager also gets the number of seconds the user has been inactive.

### 3.2.3 Attention Sensor

The attention sensor is a physical sensor based on a webcam associated with a given installation of the MyUI system. The webcam application running on the same device the MyUI interface is running on detects the line of vision of a user by tracking his eyes, nose and mouth. If the application detects an interruption of the line of vision between the webcam and the end-user that lasts longer than a configurable threshold a sensor event is issued stating that the user is inattentive.

The current implementation of the Context Manager adopts the user profile variable for attention once such a sensor event arrives. It has to be tested if it is reasonable to include further analysis of previous sensor events stating the inattentiveness of a user since a single event may not mean that the user is inattentive in general. However many of these events in a short period of time could give better evidence of the inattentiveness of the user.

### 3.2.4 TmtGameSensor

In order to support the games developed in task 4.5 of WP4 a virtual sensor associated with a given user was added to capture the results the end-user achieved while playing the Trail-Making-Test

game. This concept of a virtual sensor has two different properties containing the game results. First the time it takes the user to complete the game is measured, second the number of misclicks the user produced while playing the game is measured.

Currently these measurements are used to adopt the user profile variables for attention, processing speed, working memory and finger precision. For further details please refer to the section dedicated to the games in this deliverable and to the upcoming deliverable D4.5 which will present the games and the user profile changes.

### 3.2.5 CorsiTestSensor

As with the TmtGameSensor, the CorsiTestSensor has been integrated into the ontology in order to support the games developed in task 4.5 of WP4. The CorsiTestSensor is a virtual sensor associated with a given user that is used to represent the results of the Corsi-Test game. The CorsiTestSensor has two different properties. The first contains the highest game-level the end-user reached while the second contains the final score the end-user achieved.

### 3.2.6 Current State of the Sensor Ontology

As stated several new concepts have been introduced to the MyUI sensor ontology. In order to get a complete overview of this ontology the following graphical representation has been prepared:

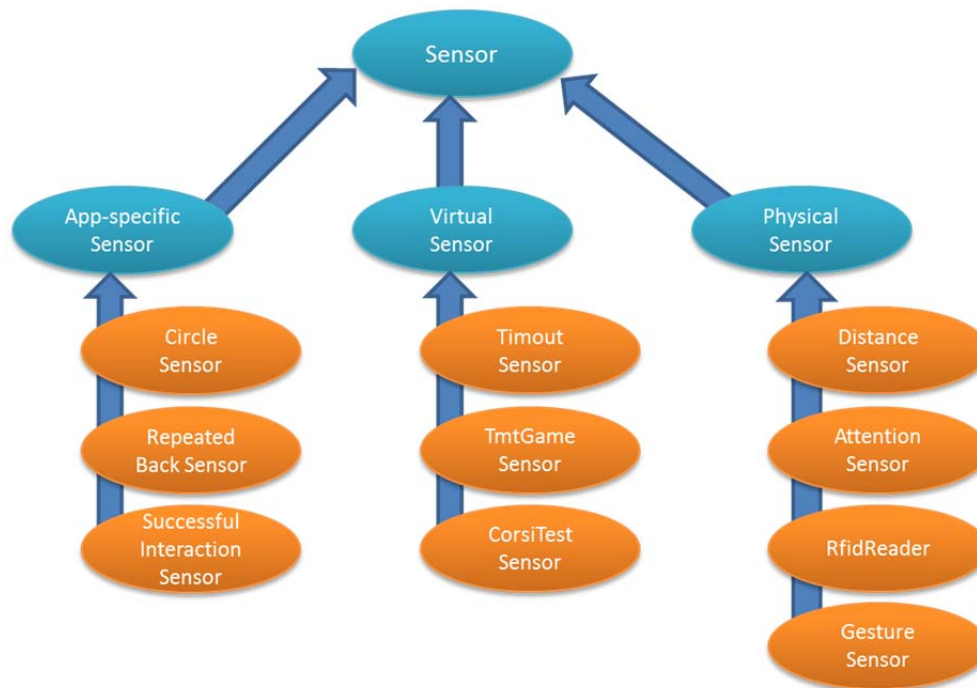


Figure 11: Current state of the Sensor Ontology

## 4. Finalization of the User Profile

### 4.1 Final Version

This chapter presents the final version of the MyUI user profile. While most of the user profile variables have been kept since D1.1 some have been added or removed. The subchapters below Table 1 provide detailed descriptions of the new user profile variables.

Name of variable	Description	Value Space	Taxonomy	Data Type	Reference / Source
Visual Acuity	Ability to perceive what is displayed on the screen	[0,4]	Perceptual	Float	WHO ICF B2100 WHO ICF B21022 WHO ICF B21020 ISO 22411 D2.1: URS07 D2.1: URS09 FRS01-02
Field of Vision	Describes how limited the field of vision of the given user is.	[0,4]	Perceptual	Float	
Ambient Light	The amount of ambient light at the users place.	[0,4]	Perceptual	Float	
Ambient Noise	The amount of ambient noise at the users place.	[0,4]	Perceptual	Float	
Hearing	Describes how limited the user's ability to hear sounds is.	[0,4]	Perceptual	Float	
Language Reception	Ability to understand written or spoken language	[0,4]	Cognitive	Float	WHO ICF B1670 ISO 22411
Language Production	Ability to speak and write language	[0,4]	Cognitive	Float	WHO ICF B1671 ISO 22411 D2.1: URC07
Understanding Abstract Signs	Ability to understand abstract signs and pictograms	[0,4]	Cognitive	Float	D2.1: UR20
Attention	Ability to handle multiple things at the same time, resp. focusing on something.	[0,4]	Cognitive	Float	D2.1: URC05 WHO ICF B140 ISO 22411
Processing Speed	Ability to process information fast.	[0,4]	Cognitive	Float	D2.1: URC04 ISO 22411
Working Memory	Ability to remember an exact sequence of steps in a process and the ability to orientate in this process.	[0,4]	Cognitive	Float	D2.1: URC03 ISO 22411
Long Term Memory	Ability to learn and remember information for a long time.	[0,4]	Cognitive	Float	D2.1: URC02
ICT Literacy	Ability to use modern	[0,4]	Cognitive	Float	

	information technology.				
Hand-Eye Coordination	Ability to coordinate the movement of the hands with things seen.	[0,4]	Cognitive	Float	D2.1: URC19
Speech Articulation	Ability to speak	[0,4]	Motor	Float	WHO ICF B310
Finger Precision	Ability to move the fingers precisely.	[0,4]	Motor	Float	ISO 22411
Hand Precision	Ability to move the hand precisely.	[0,4]	Motor	Float	ISO 22411
Arm Precision	Ability to move the arms precisely.	[0,4]	Motor	Float	ISO 22411
Contact Grip	Ability to control things by touching them.	[0,4]	Motor	Float	ISO 22411
Pinch Grip	Ability to press single buttons.	[0,4]	Motor	Float	ISO 22411
Clench Grip	Ability to hold object.	[0,4]	Motor	Float	ISO 22411
First Name	The first name of the user.		General	String	
Last Name	The last name of the user.		General	String	
Email Address	The email address of the user.		General	String	
Preferred Language	The language the user prefers to use.	English, German, Spanish	General	Enumeration	
Successful Interactions	The number of successful interactions with the system.		General	Integer	D1.2
State transitions	The number of state transitions the user carried out.		General	Integer	D1.2
MyUI Experience	The experience with the MyUI system.	[0, 4]	General	Float	D1.2
PreferenceTonalOutput	Selects whether the user prefers output enhanced with sounds.	True False	General	Boolean	
PreferenceSpeechOutput	Selects whether the user prefers speech-output in addition to text.	True False	General	Boolean	

Table 1: Final MyUI user profile

#### 4.1.1 User Profile Variable: Successful Interactions

As shown in the section which describes the development of an instance of the user profile over time, the number of successful interactions needs to be saved inside the user profile. In order to do this a new user profile variable has been integrated into the user profile which is stored as an integer data type.

#### 4.1.2 User Profile Variable: State transitions

To facilitate the measurement of experience with the MyUI framework for a given user the user profile has been extended to include the number of state transitions the user carried out in the MyUI system. This can be regarded as a measurement for the usage time of the system by the given user because every usage of the system is connected to state transitions.

#### 4.1.3 User Profile Variable: Experience

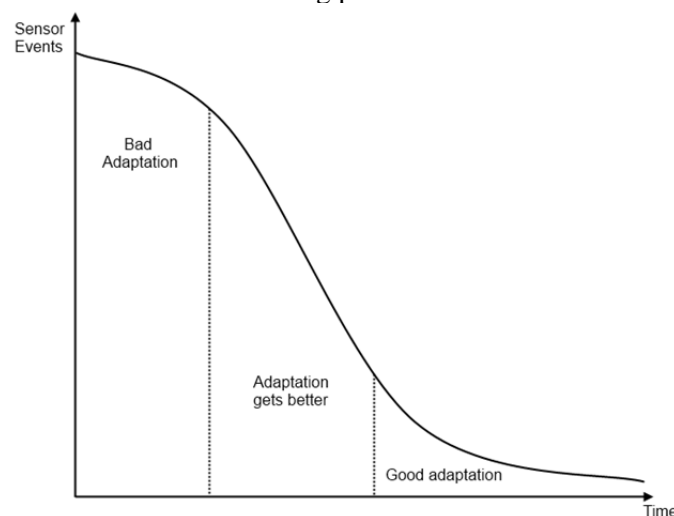
What is the definition of experience? Perhaps some function of usage time and number of successful interactions. A high usage time alone does not mean that the user has a high level of experience with the MyUI system. Furthermore some sort of gradient for the successful interactions over time could lead to some results. A steep gradient would indicate that the user is currently performing quite good which is only possible with a certain amount of experience with the system and a good adaptation of the user interface. A small gradient would either mean that the user has not much experience with the system and still learns how to use it, or that the adaptation of the user interface is not suited for the user.

### 4.2 Development of an Instance of the User Profile

The initial user profile does not necessarily have something to do with the capabilities and limitations of the user. It starts with all values set to zero which means that the user has no limitations regarding the usage of ICT user interfaces.

Since the difference between the actual value of a user profile variable and the reality can be large the user will have problems to use the user interface adapted to the initial instance of the user profile. Another problem is that the user does not know the system and will therefore make mistakes during usage. This means that there will be many sensor measurements indicating these problems arriving at the Context Manager.

As the Context Manager is adapting the user profile in accordance with the arriving sensor events the user profile will start fitting the capabilities and limitations of the user better. The better the user profile gets the less sensor events indicating problems with the user interface will occur.



**Figure 12: Development of a user profile over time.**

At one point the user interface adaptation and the knowledge of the user about the system reaches a point where further adaptations won't make any sense since the user can handle the system without any usability problems. However the sensors will still perform measurements of the user and there is a chance that sensors provide false-positives about possible problems the user has during the usage of the MyUI system. In the worst case these measurements trigger changes in the user profile which result in an adaptation of the user interface. The user's limitations are rated worse than they actually are. This behavior can therefore lead to a worsened user experience.

#### 4.2.1 User Profile Drift

The situation described above is inherent to the MyUI system. Since the user profile and the sensor measurements focus only on the detection of problems in the interaction between the user and the user interface, every sensor event makes the value of user profile variables worse. This leads to all user profile variables capturing the limitations of a user converging against the maximum value of four over time.

There are multiple possibilities to resolve this issue which are discussed in the following sections.

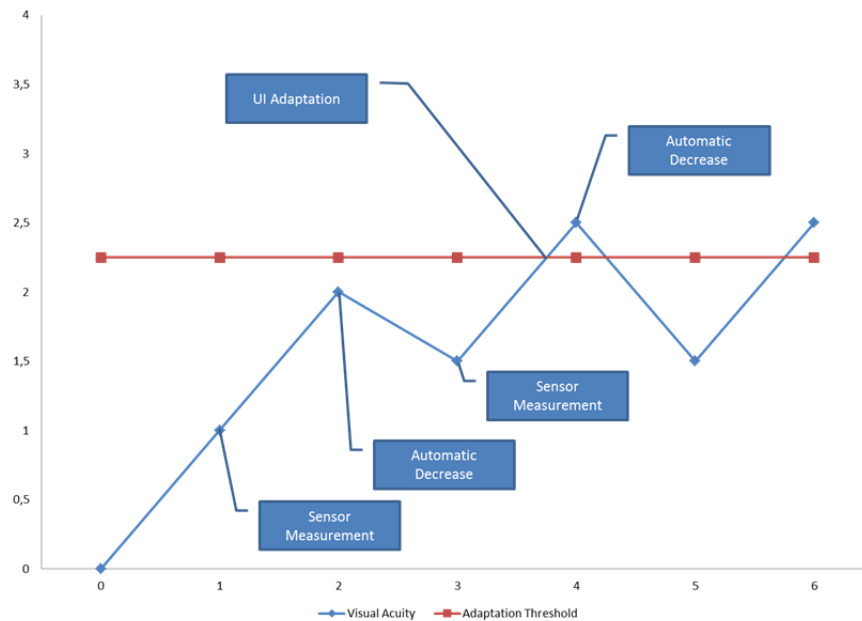
##### 4.2.1.1 *Explicit Continuous Assessment of Users*

The user is assessed with a test battery on a regular schedule in order to measure the quality of the adaptation. Although this method would provide the best adaptation to the user there are several drawbacks that would need to be investigated.

The first problem would be that there has to be at least one explicit assessment for each user profile variable. Although the neuropsychology provides us with a broad range of standardized assessment tools for a large number of limitations and capabilities (see Lezak et.al.) the compilation of such a complex test battery would consume a lot of time and would have to be done with every change of the user profile structure in the future. Also the user would have to complete the whole test battery on a regular base in order to keep his user profile up-to-date in the MyUI system. The completion of such a large quantity of tests would consume a lot of time on the side of the user. Given the fact that one of the goals of MyUI is to be mostly unobtrusive with regards to the user such methods cannot be regarded as a solution to the problem of negative user profile drift.

##### 4.2.1.2 *Fixed Continuous Decrease of User Profile Variables*

Another possibility to handle the negative user profile drift is to continuously decrease the user profile variables of a given user. During the usage of the MyUI system all user profile variables capturing limitations of a certain user will be continuously decreased by a constant factor. The decrease will occur at fixed time intervals.



**Figure 13: Qualitative illustration of fixed continuous decrement of user profile variables**

The diagram above illustrates an example for this solution. The blue line indicates the value of the user profile variable “Visual Acuity” at time  $t=0$  to  $t=6$ . The red line is the threshold where the user interface adaptation will change based on “Visual Acuity”. At time  $t=0$  the user profile is initialized for the first time.

When looking at the illustration above it becomes clear that this approach can lead to a system that will constantly toggle at the transition between two different UI adaptations. For example “Visual Acuity” is set to 2.9 and a sensor event arrives which leads to “Visual Acuity” being set to 3.4 the user interface will be adapted. If the factor is 0.5 the user profile variable will be set back to 2.9 after a certain amount of time if no other sensor event arrives. This again leads to an adaptation of the user interface.

It is also being mentioned that the time interval between two automatic decreases of the user profile variables plays an important role. If this time interval is configured to be too small then the user profile will constantly describe the user as being better than he actually is. This becomes the case when the user generates less sensor events than automatic decreases happen. If the time interval is too large the user profile drift will be slowed down but not completely eliminated. The same assumptions hold when it comes to the factor by which the user profile is decreased every time the automatic decrease takes place.

#### 4.2.1.3 False-Positive Detection on Update

Since the abovementioned method raises the problem of defining exactly the time interval between decrements and the value of the decrement of the user profile variables another method will be proposed. While the fixed continuous decrease method for the user profile variables is an “antagonist” to the sensor events - which increment user profile variables – that works more or less independent from the sensor events, the false-positive detection tries to adapt the user profile changes caused by sensor events directly.

This adaptation will be based on the Context Manager deciding whether each user profile change will lead to a positive change in the UI adaptation or if the change is a false-positive. A false-positive would be a sensor event which results in the change of one or more user profile variables which does not adapt the user interface in a way that increases usability or – even worse – reduces usability for the user.

This would mean for a perfectly adapted user interface the rate of detected sensor events during its usage would converge against zero, while the number of successful interactions between the user and the interface would be high. For a poor adaptation to the needs of the user the opposite would be the case: while the rate of detected sensor events would be very high, the rate of successful interactions would be low.

To evaluate whether an adaptation is good or bad the ratio between the number of sensor events and the number of successful interactions could be used.

### 4.3 Manual Adaptation of the User Profile

Depending on the usage frequency and the intensity of MyUI it takes some time until the system is adapted to the user in a way that usage is possible without problems. During this training period it is difficult to learn how to use the system because of multiple adaptations of the user interface taking place.

To overcome this issue an initial assessment of the user at the beginning of his MyUI usage would be necessary, where an acceptable adaptation is achieved by directly setting the values of user profile variables. This assessment could be done by specially trained personal during a bilateral session with the user.

The basic assumption behind this is that such a manually created user profile will be better suited for the user than a user profile which is solely based on automated measurements.

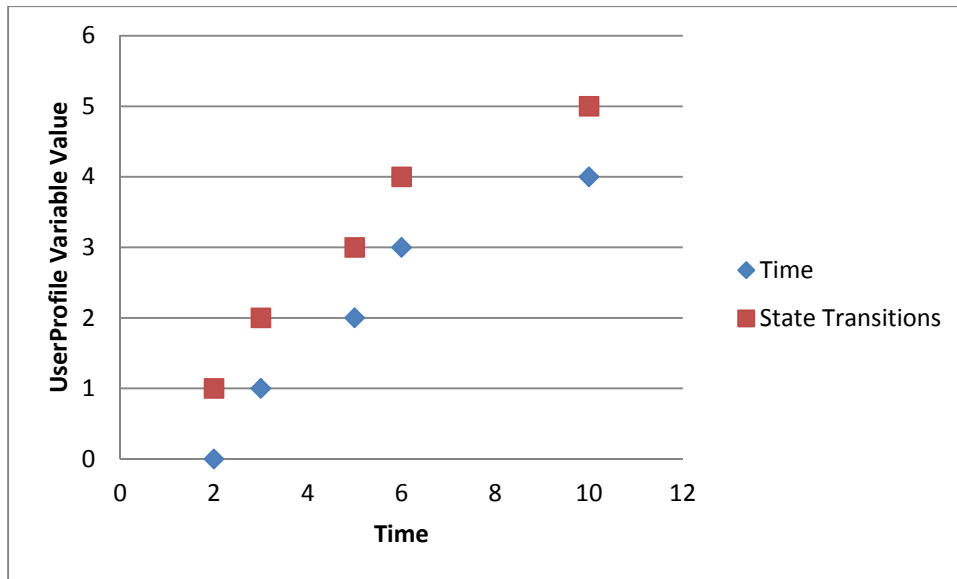
Although the user profile created during the manual assessment of the user should provide a reasonable good adaptation of the user interface it is still possible that the user interface should adapt to the user. This happens because

- the performance of the user during the usage of applications is not constant with regards to “good” and “bad” days of the user
- the sensors connected to the MyUI system measure false-positives indicating a limitation of the capabilities of a user although no limitation exists. (see page 23)

With the possibility of false-positives and their detection already covered in a subsequent chapter of this deliverable there is also a proposed solution that covers the variation of user capabilities and limitations.

While a manually set value of a user profile variable provides a very good mapping of the user’s capabilities and limitations at the time of setting the variable value, the value can still change over time. However one would not expect the user profile variable value to change largely. In fact there would be small derivations from the manually set value. This would mean that we would have to put a smaller weight on the user profile variable updates coming from sensor events after a user profile variable has been set.





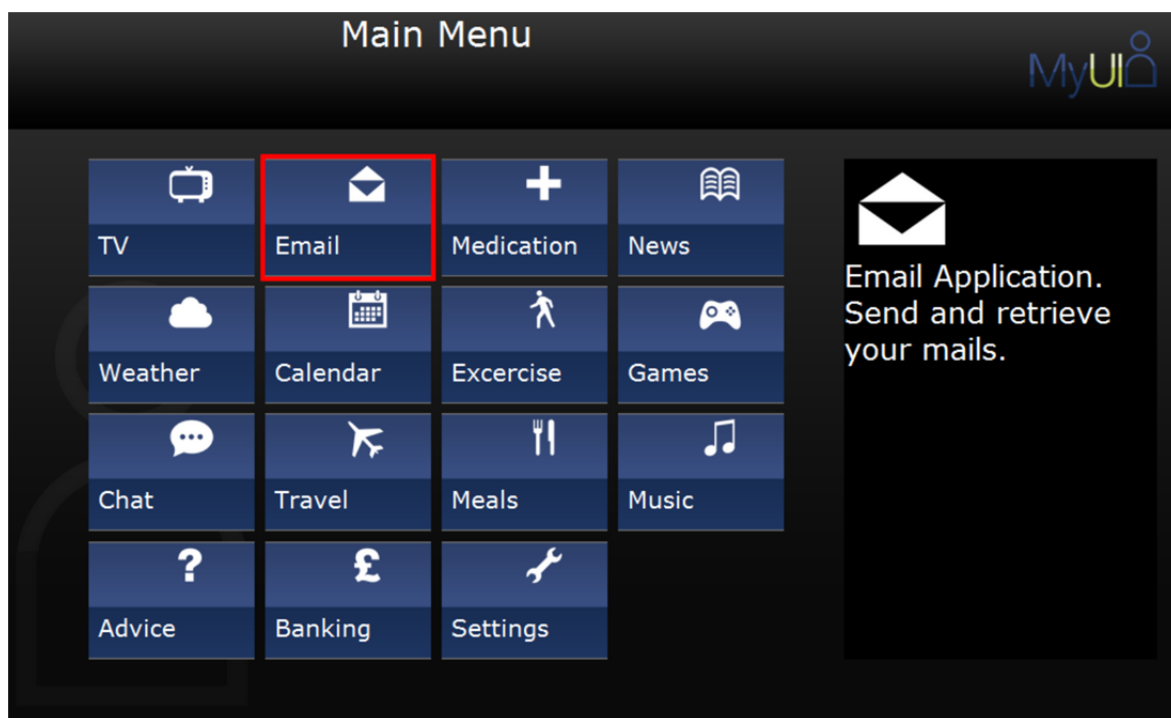
**Figure 14: Example of a explicitly set user profile variable (at t=1) with subsequent sensor events (t=2 and t=3). The red squares indicate the new value with a weight for the sensor events of 0.2 and the blue diamonds indicate the new values without a weight.**

For example if a user profile variable has been explicitly set to 2, subsequent sensor events could change the value of a user profile variable in a way that the derivation from the explicitly set value becomes very large. This would be inaccurate since we assume that the explicitly set value is a better representation of the user. Therefore a factor of 0.2 is applied to changes of user profile variables that have been set explicitly before if the changes originate in a sensor event.

## 5. The Role of Games for User Profiling

Finally, this section briefly reviews the new role of the Context Manager in providing an initial estimate of the User Profile. Essentially, a set of basic games, offered to the new user, will provide a basic estimate on the user's perceptual, motor and cognitive skills, which will serve to aid the User Interface Adaptation Engine in a first User Interface proposal. Next we provide a first version of the current games developed in Task T4.5 of Work Package 4 for a first estimate on the user's profile, but the reader is referred to deliverable D4.5 for a more detailed documentation on this matter.

Figure 15 shows the welcome menu for a given user. As shown, second row, fourth column offers the application Games, where these games may be accessed by the user.



**Figure 15: Main menu for a user without impairments**

All games have been developed in Flash, and they easily integrate with the CakePHP framework.

*Game: Trail Making Test.*

Figure 16 shows the welcome page for the Trail Making Test Game, where the user can either start the game or read the Instructions of this game.



Figure 16: Welcome screen of the Trail Making Test game

After clicking Start, the user will be prompted with the game shown in Figure 17.

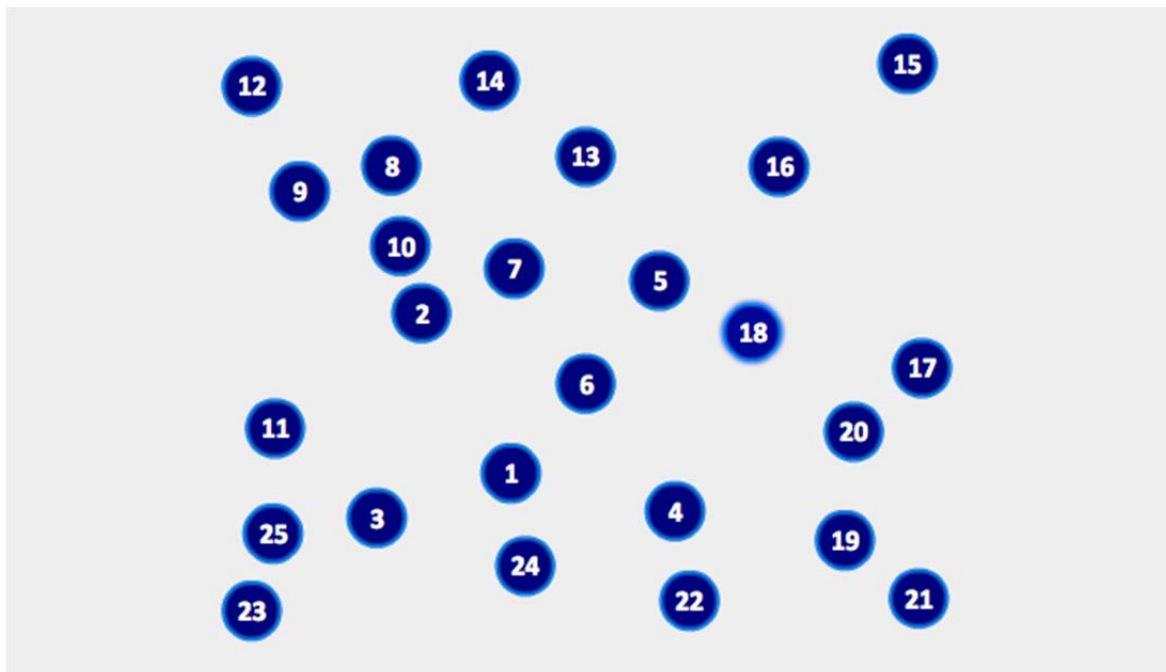


Figure 17: The Trail Making Test game

In this game, the user must click on the circles following an increasing order, that is, the user must find the circle number 1 and click it, then find circle number 2 and click it, the circle number 3 and so on until he/she clicks the 25 circles. The program measures the amount of time that the user takes in completing this game and the number of wrong clicks, both out of sequence clicks and clicks outside the circles (see Fig. 13). These three parameters capture several cognitive and motor skills of the user, and can be used to establish a first estimate on some user's profile variables, for instance:

- Attention, which is a numeric value in the range [0,4]
- ProcessingSpeed, again a numeric value in the range [0,4]
- WorkingMemory, numeric value in the range [0,4]
- FingerPrecision, numeric value in the range [0,4]

Essentially, the time taken by the user to complete the game should provide input to Attention, ProcessingSpeed and WorkingMemory. However, the number of wrong clicks may produce an update on Attention and FingerPrecision. The matching between the game scores and the Context Manager attributes will be further refined in Deliverable D4.5.

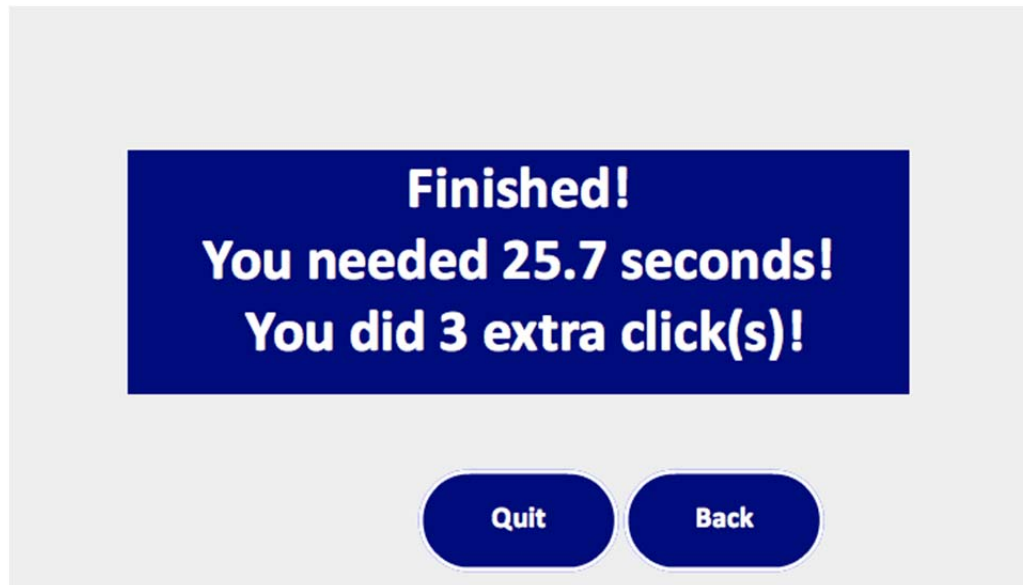


Figure 18: Output of the Trail Making Test game after completion

## 6. Summary and discussion

This deliverable has reviewed the updated version of the Context Manager (CM) from the original version described in D1.1. Conceptually, the Context Manager remains the same but it has been refined with new attributes collected from the WebCam and the RemoteController devices. These sensors provide input about the user concerning aspects: Perceptual, Cognitive and Motor skills of the user. Such new information provides the base for highly individualised User Interface to a particular user.

In addition, the CM has been reimplemented in C# because of its better performance behaviour and code simplicity. The new Context Manager also provides a Graphical User Interface (GUI) that eases the management and administration of the sensors, users, and attributes involved in the User Profile. A summary of this GUI has been revised throughout the document.

Finally, this deliverable also describes the new role of the Context Manager in capturing an initial profile of the user via cognitive games. This new role of the Context Manager is further extended in D4.5 from WP4.

## References

1. A. 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.
2. S. Staab and R. Studer: "Handbook on Ontologies". Second edition. Springer, 2009.
3. S. Rollwage Kresser, M. Klein and P. Wolf "Collaborating context reasoners as basis for affordable AAL Systems. In *Proc. of the 4rd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI09)*, 2009.
4. P. Wolf and A. Schmidt and M. Klein: "Applying Semantic Technologies for Context-Aware AAL Services: What we can learn from SOPRANO". In *Proc. of Workshop on Applications of Semantic Technologies, Informatik 2009*
5. P. Wolf, A. Schmidt, J. Parada Otte, M. Klein, S. Rollwage, B. König-Ries, T. Dettborn, A. Gabdulkhakova: "OpenAAL - the open source middleware for ambient-assisted living (AAL)". In *Proc. of AALIANCE conference, Malaga, Spain, March 11-12, 2010*