Project Deliverable Report

Deliverable 3.2 – Definition and Implementation of the Conceptual Model for Context Awareness in idSpace v1

Work Package 3
Task 3.4
Date of delivery Contractual: 31-12-2008       Actual: 16-02-2009
Code name Version: 0.0 Draft ☐ Final ☑
Type of deliverable Report
Security (distribution level) PU
Contributors
Authors (Partner) George Sielis (UCY), Aimilia Tzanavari (UCY), Constantinos Kakousis (UCY), Nearchos Paspallis (UCY), George A. Papadopoulos (UCY)
Contact Person George Sielis, Aimilia Tzanavari
WP/Task responsible WP3
EC Project Officer Mr. Christian Wilk

Abstract (for dissemination) This deliverable defines the methods of implementation of the conceptual model for the first version of the Context Awareness prototype for idSpace. It defines the architecture of the prototype and describes the parts of the context component. Samples of code are given, explaining the overall structure of the prototype and the partial use of the component in the individual architectural layers.

Keywords List Context awareness, prototype, context reasoning, adaptation reasoning, context manager

idSpace Project Coordination at: Open University of the Netherlands
Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands
Tel: +31 45 5762624  Fax: +31 45 5762800
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>7</td>
</tr>
<tr>
<td>1.1</td>
<td>Background</td>
<td>7</td>
</tr>
<tr>
<td>1.2</td>
<td>Context Awareness Overview</td>
<td>8</td>
</tr>
<tr>
<td>1.3</td>
<td>Problem Statement and Research Objectives</td>
<td>8</td>
</tr>
<tr>
<td>1.4</td>
<td>Relation to Other Deliverables</td>
<td>8</td>
</tr>
<tr>
<td>1.5</td>
<td>Structure of this Deliverable</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>Contextual Reasoning</td>
<td>10</td>
</tr>
<tr>
<td>2.1</td>
<td>The Case of idSpace</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>idSpace Context Awareness Component - Architecture overview</td>
<td>12</td>
</tr>
<tr>
<td>3.1</td>
<td>Context Manager</td>
<td>18</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Validator</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>Wrapper</td>
<td>22</td>
</tr>
<tr>
<td>3.3</td>
<td>Adaptation Manager</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>Prototype</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>Conclusion and future work</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>References</td>
<td>30</td>
</tr>
</tbody>
</table>

Appendix A | Functional Architecture | 34 |
| A.1       | Reasoning Techniques | 36 |
| A.2.1     | Non-symbolic Reasoning Techniques | 37 |
| A.2.2     | Symbolic Reasoning Techniques | 37 |
| A.2.3     | Hybrid Reasoning Techniques | 40 |
| A.3       | Technologies for Context Reasoning | 40 |
| A.3.1     | Context abstraction using rule-based reasoning engines | 40 |
| A.3.2     | Ontology-based inference engines | 42 |
| A.3.3     | Topic Map Technologies | 43 |
| A.4       | Technologies for Context Storage and Retrieval | 46 |
| A.4.1     | Existing technologies for context storage and retrieval | 46 |

Appendix B | Java Script function for Browsing with AJAX technology | 49 |
| B.1       | Java Script Functions for collecting System Information from the Client | 50 |
| B.3       | Java Script for the creation of XML file with the System’s data | 52 |
B.4 Example of the created XML file with the System’s data ........................................... 52
B.5 XML Schema for idSpace example ........................................................................... 53
B.6 XML Context Data Example ....................................................................................... 58
B.7 XML Validator in PHP ................................................................................................ 59
B.8 PHP reading / writing XML data files ....................................................................... 60
# TABLE OF FIGURES AND TABLES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Context Awareness Ontology</td>
<td>7</td>
</tr>
<tr>
<td>Figure 2</td>
<td>idSpace Platform Architecture [D4.1]</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Context Awareness in idSpace</td>
<td>13</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Context Awareness Component Concept</td>
<td>14</td>
</tr>
<tr>
<td>Figure 5</td>
<td>idSpace Context Component Architecture</td>
<td>14</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Steps of Context Component Functioning</td>
<td>16</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Database Schema Example</td>
<td>19</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Binary Representation of Context Entities</td>
<td>20</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Example of XML data document</td>
<td>21</td>
</tr>
<tr>
<td>Figure 10</td>
<td>XML Validation in PHP</td>
<td>22</td>
</tr>
<tr>
<td>Figure 11</td>
<td>XSLT example</td>
<td>23</td>
</tr>
<tr>
<td>Figure 12</td>
<td>xpath() example</td>
<td>23</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Example of the Context Awareness Prototype</td>
<td>28</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Reasoning Engine Functional Stages</td>
<td>35</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Functional Architecture of Context Reasoning Engine</td>
<td>35</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Example of Symbolic Simulation (Csallner et al., 2008)</td>
<td>38</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

1.1 Background

This deliverable, the second one to be produced within Work Package 3, aims to describe how context awareness will be implemented in the idSpace platform. Context Awareness for idSpace was initially described in Deliverable D3.1 following an extended State of the Art Analysis. The main objective of that first deliverable was to give a clear definition of Context Awareness for idSpace. With this definition, it became possible to point out the context elements, and design the conceptual model for the idSpace Context Awareness Ontology.

This document continues the study of Context Awareness for the idSpace platform and its conceptual model. The Context Awareness Ontology (Figure 1) and the contextual elements User, Social Environment, Task and System, are the starting points for an extended definition of the conceptual model and its implementation, which will be presented in the chapters to follow.

Figure 1: Context Awareness Ontology
1.2 Context Awareness Overview

Context Awareness will be developed as a framework component of the idSpace platform which will be responsible to describe the entities defined in the context ontology model. The entities may be characterized as users, actions, or the progress status of a procedure within the platform. Context awareness aims to enhance the creativity process by the recommendation of people for collaboration, recommendation of resources related to the objective of an idea, recommendation of creativity technique to be used in a particular situation, recommendation of an alternative conceptual view of an idea, etc. The implementation of the aforementioned is supported by the semantic modeling of the context elements within the Context Awareness Ontology (Figure 1). The contextual elements will be characterized by attributes that will be subsequently used for the determination of how and when the Context Awareness component will be used.

The idSpace platform is an ontology-based web application and therefore Context Awareness will be modeled as a separate ontology in the platform.

1.3 Problem Statement and Research Objectives

The objective of this document is to present the first version of the definition of the conceptual model and the implementation of Context Awareness for idSpace. The conceptual model of the context awareness will be described through schemes and functions, which will explain how the context awareness component will be implemented and adjusted to idSpace. To define the conceptual model we will approach the problem by firstly presenting related work in the areas of Contextual Reasoning Techniques, Technologies for Context Reasoning and Technologies for Context Storage and Retrieval.

1.4 Relation to Other Deliverables

The Deliverable is related to the following deliverables:

- **D1.1 State of the Art on Pedagogical Strategies.** This deliverable focuses on support to enhance creativity via pedagogical strategies. The State of the art on the definition of the context factors needed to be taken into account when creating affording circumstances for creativity is the common ground between D1.1 and this Deliverable. By knowing these context factors and by knowing how these factors should be used by the context component guide the structured planning and designing of the context awareness prototype.

- **D2.1. State of the Art in Tools for Creativity** provides a survey of creativity techniques that have been developed over time. It provides a reference model for characterizing them as a basis for selecting techniques that are particularly appropriate for the idSpace project. The deliverable also surveys existing creativity tools and describes them in terms of the underlying techniques. Through the State of the Art in Creativity tools, the parameters and characteristics of the existing creativity tech-
niques are important for pointing out how this information can be transferred to context information. Also, the description of the existing creativity tools is emphasizing the need for the existence of context awareness for the creation of enhanced collaborative learning.

- **D4.1 State of the Art in Tools for Creativity** describes the idSpace platform architecture and the position of Context Awareness in the platform, the technologies used for the implementation and the integration of Context Awareness in the platform. D4.1 is the most related document to this deliverable. The description of the platforms and the technologies used for their integration, are setting the restrictions and the limitations on the development of a new component. A new framework component like the context awareness component has to be compatible with the technologies used for the platform’s integration, be aware of the data storage methods and be aware of the level of interoperability of the platform. Additionally, the methods of interaction and communication with other technologies and platforms, which are also covered in D4.1, are important for the context component.

### 1.5 Structure of this Deliverable

The deliverable is structured as follows. Chapter 2 presents our choice of technologies for the implementation of context awareness. The reader interested in an overview of Contextual Reasoning, referring to reasoning techniques, technologies for context reasoning, as well as technologies for context storage and retrieval is referred to Appendix A. Chapter 3 includes an overview of the idSpace Context Awareness component architecture, explaining the parts involved and their functionality. Chapter 4 describes the derived prototype and the work carried out for this initial version’s implementation. Chapter 5 summarizes the overall document and briefly describes the future work.
2 CONTEXTUAL REASONING

When taking a formal approach to model context, context can be processed with logical reasoning mechanisms and has two approaches: checking the consistency of context, and deducing high-level, implicit context from low-level, explicit context (Wang, Zhang, Gu, & Pung, 2004). Modeling context as an individual ontology offers the possibility to declare and control the reasoning mechanisms. Contextual reasoning refers to the creation of relations and associations that contribute as the reasoning rules for the extraction of the contextual information. Giunchiglia, (1993) gives the meaning of contextual reasoning by describing the difference between context and situation. According to Giunchiglia, (1993), contexts are not situations. A situation records the state of the world as it is, independently of how it is represented in the mind of the reasoner. On the other hand context is inside the reasoning individual. It is part of the state and it is responsible for the subjective view of the captured information.

In computing systems the formal representation of context helps in capturing the information, which is characterized by a situation, and then set the reasoning mechanism for the creation of the context.

Context Awareness research has developed reasoning techniques and technologies for context reasoning. A more specialized literature review on the functional architecture of contextual reasoning, reasoning techniques, technologies for context reasoning, as well as technologies for context storage and retrieval can be found in Appendix A. The techniques and technologies reviewed, formed the basis for the idSpace context awareness component architecture and implementation.

2.1 The Case of idSpace

From the restrictions for the technologies used in idSpace and from a consideration of the overall usage scenarios as the objective tasks of the project, a proposal for context reasoning techniques, adaptation reasoning techniques and technologies for context reasoning follow. For a detailed argumentation, see Appendix A.

Given the use of ontologies in the idSpace platform and the existence of an individual ontology for the context awareness component, it becomes clear that the appropriate context reasoning techniques for our case would belong to the group of Symbolic Reasoning Techniques. The use of Non Symbolic or Hybrid context reasoning techniques is not within the scope of the project. Furthermore, the use of Artificial Intelligence and Machine Learning algorithms would highly increase the complexity of the context awareness component.

Among the various context reasoning techniques described in section A.3, we believe that action-based reasoning, enhanced with fuzzy logic, is the most suitable approach for the case of idSpace. On one hand, we have deterministic context space (i.e. we only allow predefined context events to occur during the system execution) giving us the capability of modeling our context space in rules and actions. On the other hand, introducing fuzziness to our reasoning enables for multiple valid adaptation suggestions which can be prioritized according
to their fitness to the current context. Although action-based techniques require deep knowledge from the system developer they precisely define how the system should react after a context change occurs. In addition, it is the easiest and most commonly used adaptation reasoning technique which we believe that, in combination with fuzzy logic, can successfully demonstrate the adaptation behavior of the idSpace platform. Alternative solutions such as goal-based policies and AI techniques, although they can provide sophisticated adaptation behavior they are considered too complicated for the context awareness purposes of this project.

For the purposes of the initial prototype presented in this Deliverable will not use one of the existing engines described in Section A.4 for context storage and retrieval. For the implementation of this prototype, available technologies will be used to manage the storage and retrieval of context manually.

The idSpace project has its own restrictions with respect to the technologies used. As it will be described in the following sections idSpace is an ontology based web application, which is using Topic Maps technology (Anonymous, 2008c) and its front end API is implemented in PHP (Anonymous, 2009c). Considering the restrictions and the selected methods and techniques the context awareness prototype will be developed and implemented as it is presented in the following sections.
3 IDSPACE CONTEXT AWARENESS COMPONENT - ARCHITECTURE OVERVIEW

Based on the functional Architecture as it is described in section A.2 and considering the technologies idSpace uses and architecture given in Deliverable D4.1, this section describes the architecture of the context awareness framework component for idSpace.

Context awareness for idSpace will be implemented as an individual component. The component will be adjusted to the platform and it will be able to apply its functions based on the usage scenarios. The overall architecture of the idSpace platform presented in D4.1 is shown in Figure 2.

As shown in Figure 2, Context Awareness belongs to the internal Logic Modules of the platform. The idSpace platform is a web-based application which involves two subsystems: the Microcosmos Content Management System and the Kamala Topic Maps tool which is based on the Ontopia Topic Maps engine. The two sub-systems are communicating with web services protocols. Context Awareness will be applied as an internal Logic component of Microcosmos platform but it will also be part of the kamala engine. Kamala will contain the idSpace ontologies, one of which is the Context awareness ontology (see Figure 1).

Figure 2: idSpace Platform Architecture [D4.1]
The integration language of the idSpace platform is the PHP scripting language for web applications. Thus the context awareness component must be implemented with the same language using semantics for data control through the Topic Maps engine.

The context awareness ontology as it is described in Deliverable D3.1 contains the context elements defined for the idSpace platform. The fact that context awareness for idSpace must be implemented for a web application limits the options of using hardware sensors for tracking events and actions. The solution for its implementation was found in the creation of the ontology using a combination of reasoning techniques and semantic web advantages. An important task of the context awareness component will be the user’s recognition. This recognition will be achieved by tracking the user’s progress status and subsequently sending the server the necessary client information. On the server side, a reasoning engine will be responsible for creating queries and adaptation methods that will then create the recommended information data.

The implementation of the context awareness component will be done using the PHP scripting language in combination with XML markup language.

For the design of the flow of the context awareness component, the usage scenarios were considered. According to the scenarios the component must keep track of the users’ actions for two options: 1. user as individual system unit and 2. user as participant in a session, collaborating with other users.

The users of the platform will be able to work single or participating in sessions and collaborating with others. In both cases when a user is logging into the system a client side script will be activated and collect all the system data of the user’s system attributes. This information will be tracked in a preformed XML file. This file will be sending the information to the server and save it to the database. While the user is navigating to the system a personal log file will be tracking his actions. A similar log file will exist for a session, which will track the actions of all the participants of the session. With this method the context component will be able to control the actions and the events coming from the user or by a group of users during an idSpace session process.

An overview of the context awareness component is shown in Figure 4.
The overall architecture of the proposed prototype for the context awareness engine for idSpace is based on the context awareness functional architecture presented in Section A.1 of Appendix A (Figure 15). The idSpace context engine will be implemented in 3 layers – subsystems.

1. Context Manager
2. Wrapper
3. Adaptation Manager

The architecture scheme is shown in figure 5.
Each layer will be implemented as a separate engine. Each engine has specified tasks and roles for the context awareness component functioning and for this purpose the interaction between them is mandatory.

The Context Manager will be responsible to capture the predefined events and map the action events coming from the user to the context component function events for capturing, filtering and representing the requested information. The Context Manager will contain the validation functions. The Validator within the Context Manager will be responsible to validate the input and the retrieved data types based on an XML scheme extracted by the Topic Maps engine.

The Wrapper will capture the requested information in an XML file. This data file will be used by the Adaptation Reasoning to query the data and present them to the user based on the predefined context rules.

A database engine will be used for the context data storage and retrieval. The database tables will be designed based on the entities and their relations-associations defined in the context awareness ontology.

The idSpace context component architecture as it is shown in Figure 5 will be implemented based on the technologies and the restrictions used from the overall architecture of the idSpace platform. The use of Topic Maps technology and PHP as the front-end implementation language, are important factors that should be considered for the implementation of the context awareness component and its integration within the platform.

The architecture levels of the component and the implementation methods will be described in the following sections in more detail with coding examples. The proposed architecture is based on the overall use of context component in idSpace platform. The steps – functions that the component should support are based on the real time recommendations either by giving input to the system or automatic recommendations.

Giving input from the client for processing and retrieval of context information, but also the retrieval of the context information and its presentation to the user demands the existence of the context data Validator. If the validation returns a Boolean value true then the system is ready to capture the context information, process it based on the predefined context rules and adapt it to the user who requested it.
Figure 6: Steps of Context Component Functioning

Figure 6 demonstrates the steps of the Context Component Process. The steps are designed based on the general architecture of the Context component. The six steps are as follows:

1. At this step there is the predefined list of events which signs the activation of the Context Manager subsystem. The list of events is representing the stage of the creativitpy process and the type of context requested. Each event is predefined to be mapped to a context request. The check for the activation of an event is done through the personal log file of each user. When an event is read from the log file then the system is aware of the requested context information. The requested context information is then sent to the Context Manager.
2. The Context Manager captures the event from the predefined event list. It creates the paths for the requested context information and retrieves it from the repository. Context Manager sends the captured context data to the Validator and to the Wrapper.

3. Validation and Wrapping of the context data is the process the follows the Context Manager process. The Wrapper captures the set of data that are indicated by the Context Manager. These data are collected by the Wrapper and stored in XML formatted files. The XML formatted files are forwarded to the Validator. Through the validation process the system checks the correctness of the data types of the context data written in XML format. The validation process is established through the parsing of the captured context data from an XML schema. Wrapper and Validator both depend on the event stored into the Context Manager. (For each event from the event list, the data set wrapped is different; consequently the selected XML schema is different).

4. Adaptation Manager then receives the relevant context information and the corresponding events. Adaptation Manager initiates the Adaptation Reasoner, which is responsible for mapping context changes to fuzzy rules. The Adaptation Reasoner will decide how the system will react in response to context changes.

5. The Adaptation Manager receives results from the Topic Maps engine and prioritizes them according to the Adaptation Reasoner directions.

6. The system presents the recommendations to the user.

The flow of the Context Component can be illustrated through the following example:

**Example Situation:** The system helps the team members to formulate a problem from their perspective and discuss their formulation

**Expected Context Information:** Resources related to the problem defined, Alternative keywords / synonyms, relevant previous problem definitions

**Flow:**

1. Activate the Context Manager Sub System
2. Context Manager checks user’s personal log file and relations/associations
3. If the relation/association does not exist, then it Context Manager creates one
4. The new entry is formed from the type of Activity: Problem Definition, the problem definition and the keywords associated.
5. Context Manager creates the paths to the context information based on the association nodes
6. Wrapper reads these paths and captures the context information
7. The Wrapper filters the context data considering the event that activated the procedure
8. The Context Data are sent to the Adaptation Manager
9. Adaptation Manager uses fuzzy rules for selecting the recommendations and prioritizes them.

10. The results are presented to the user

3.1 Context Manager

The Context Manager (CM) is an important part of the context component. It captures action events to map proper service events according to the predefined logic of interaction (Lee and Jeng, 2003). The context component’s goal is to give useful recommendations based on predefined events and the relevant context in each case. Context Manager is the part of the component which will capture the events, check when and whether a recommendation is applicable and subsequently collect all the contextual information needed to ultimately send to the Adaptation Manager to decide how to proceed.

The Context Manager includes the Validator and interacts with the Wrapper when it is used. When the Context Manager captures an event coming from the user, the event is tracked by the CM and the appropriate context information to be retrieved for the particular event is captured. The retrieved context information is then transformed into XML data files. These data files along with the event itself, is sent to the Adaptation Manager. The Adaptation Manager decides which of the predefined recommendation rules should be executed, based on the event detected. According to the rules themselves and the information they require, the Adaptation Manager gets the context information needed from the XML data files or from the XTM files given by the Topic Maps engine. Subsequently the rule(s) is executed and finds the intended results (i.e. resources, users, ideas, etc.) in the appropriated database, in most cases after ‘consulting’ the Topic Maps describing them. Depending on the rule used, a corresponding Utility Factor is considered to prioritize the results from the rule’s execution, i.e. the recommendations. The Utility factor indicates the utility, or in other words the significance of a particular recommendation and thus implies its priority in the list of recommendations to be presented to the user.

3.1.1 Validator

As mentioned in previous sections the context information data are stored in a database engine and they can be manipulated with ontologies. The mapping and indexing of context data from ontologies is performed with the help of Topic Maps technologies. The Topic Maps engine used in idSpace with the help of the database storage engine used can create XML schemas. The XML schemas which are XSD files contain the data types as they are recorded in the database engine and the types of relation between the data.

The data manipulation is done with the creation of XML data files which contain data identified with names enclosed in tags. The XML files can be created on the fly using the data retrieved from the database engine or by creating XML files on the client side uploading them to the server for processing.
In any case, an XML document must be well formatted and adjusted to a defined structure. This is actually the role of the XML Validator. The Validator checks the format of an XML file. An XML file is a valid document if it respects the rules dictated by a DTD or XML Schema (XSD).

For the demonstration of the Validator’s implementation we used the usage scenarios and we created a database schema based on the context ontology of idSpace platform. From the created database schema shown in Figure 7 we extracted the XML schema used for the validation.

The database tables shown in Figure 7 are the database tables for the storage of users’ data and the related with them portfolios, ideas, roles, tasks and domains as they designed in the context ontology.

A binary representation of the entities and their relations- associations is shown in Figure 8.
Figure 8: Binary Representation of Context Entities

The XML schema code used for the prototype’s example is given in the Appendix B. It is created with the use of the MSSQL server 2005 and the Microsoft Visual Studio 2008.

An example of an XML data document is shown in Figure 9. In the XML example the data types of each tag is shown. The structure of the XML document must be valid with the XML Schema (see Appendix B).
Definition and implementation of the conceptual model for Context Awareness in idSpace v1

Figure 9: Example of XML data document

Validation is a process that takes place in the context component. If the data types contained in the context data are not correct then the Context Manager will be informed for the error and the procedure will be repeated. The validation (Figure 10) is implemented using PHP script language and with the use of PHP XML validation functions which exist in PHP libraries.
3.2 Wrapper

The Wrapper is the set of functions and commands that captures a data set and puts it in XML format file. The collected data in our case is the context related data that will be further processed. The Wrapper acts like ‘taking a photo’ of the actual data on the time an event rose. This ‘photo’ of data will be used for the selection of the necessary information, and for applying the reasoning technique(s) to reach conclusions. For the implementation of the Wrapper PHP and XML technologies are used.

An example of the Wrapper functioning is when a user pushes a button asking from the system to recommend some experts. The System will capture the data of all the expert users and put them in an XML document. The XML will be created on the fly using PHP. Then an XSLT will be created and with the use of x-paths (PHP function) and the XSLT syntax the system will return to the user the recommended experts.

```php
<?php
$xml = new DOMDocument();
$xml->load('ca2.xml');

if (!$xml->schemaValidate('DataSet1.xsd')) {
    print '<b>DOMDocument::schemaValidate() Generated Errors!</b>;
    libxml_display_errors();
}
else {
    echo "validated";
    $xmlValid=true;
}
?>
```

**Figure 10: XML Validation in PHP**
Definition and implementation of the conceptual model for Context Awareness in idSpace v1

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/"
><html>
<body>
<h2>List of Learners</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th>Name</th>
<th>Role</th>
</tr>
<xsl:for-each select="UserList/User">
<xsl:if test="user_role='Expert'">
<tr>
<td><xsl:value-of select="user_name"/></td>
<td><xsl:value-of select="user_role"/></td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Figure 11: XSLT example

Figure 11 demonstrates a sample of the XSLT code which is selecting from an XML data file all the users who have user role “Expert”. With XSLT syntax it is possible to set more filter parameters and control the presented context data.

Another way of filtering and controlling the presented context data from an XML file is through the PHP function `xpath()` (Figure 12).

```php
$xp = new domxpath($dom);
$result = $xp->query("/UserList/User/Name");
print $result[0]->firstChild->data;
```

Figure 12: xpath() example

With the `xpath()` function it is possible to set path nodes and retrieving information from a XML data file. Within the `xpath` logical expressions can also be used.
3.3 Adaptation Manager

The Adaptation Manager is responsible for making adaptation decisions according to the refined context. The Adaptation Manager contains the Adaptation Reasoning engine implemented based on the adaptation reasoning techniques. As explained in section 2.1, the adaptation reasoning technique we believe is most suitable for idSpace is action-based, also employing fuzzy rules.

The Adaptation Reasoning directly communicates with the Wrapper, the previous architecture layer. It receives from the wrapper the selected context information and tracks the context changes which also sent from the wrapper. The Adaptation Manager is using the adaptation reasoning rules to refine the context information and adapt based on the context variables which are processed.

For the purposes of this deliverable the prototype will be initially implemented in a simple “if –then-else” form. The final structure and functioning of the Adaptation Manager will eventually be based on the fuzzy rules adaptation reasoning (see section A.2.2 of Appendix A).

At this point we will try to explain the fuzzy rules adaptation reasoning with an example that we base on the usage scenarios, as were described in Deliverable D5.1 (Heider, 2008).

Example: We would like to formulate a rule that decides whether we should recommend some documents to a brainstorming group, and if so in which order, based on their relevance and availability.

The value range of the relevance value is defined according to the percentage of common words between the document and the discussed idea as it was defined by a group member.

- relevance = high [100% - 80%]
- relevance = medium [80% - 40%]
- relevance = low [40% - 10%]
- relevance = none [10% - 0%]

In the same manner the availability is defined based on how easily we can access a document:

- availability = high [The document is directly available and freely distributed in electronic format]
- availability = medium [The document is directly available in electronic format but requires subscription]
- availability = low [The document is available only in hard copy]
- availability = none [The document is not available in any format]

Based on the above a possible rule is:

IF relevance >= medium AND availability == high THEN priority = high
In the same manner we can define multiple rules that cover the whole context space and enable prioritizing multiple adaptation alternatives.

When the Adaptation Reasoning is ready then it presents the final recommendations to the user.

In the idSpace platform, context awareness should demonstrate itself, or in other words, is needed in various situations, in various forms. In order to properly define the various types of context awareness required, we consulted the related user requirements, as these were identified and expressed in Deliverable D5.1 (Heider, 2008). The document referred to three main steps/elements of Innovation that were also subsequently used to categorize user requirements. A number of user requirements involved some form of context awareness. We summarize these below, emphasizing context awareness:

1. **Preparation**

   Context Awareness supportive functions for the Preparation phase:
   
   - idSpace assistance shall **aid** team members to **formulate the problem** from their perspective and discuss their formulation
   - The system shall assist the users by **recommending appropriate methods** to systematically **clarify and explore** the initial problem statement
   - The system shall support the **identification** of the participants
   - The system shall support the **selection of synchronous and asynchronous or hybrid** work modes.
   - The system shall **provide context** and **additional material** for the ideation session

2. **Ideation Event / Creativity Session**

   Context Awareness supportive functions for the Ideation Event / Creativity Session phase:
   
   - The system shall **assist** the user in **choosing a useful method** for the ideation method
   - The system shall **support the recommendation of users** with **different roles**: moderators, participants
   - The system shall **offer a number of appropriate creativity techniques**
   - The system shall **support the recommendation of refined ideas**
   - The system shall **support learning** about the ideas of the other participants by **proposing resources related with the ideas**
   - The system shall be able to **suggest on demand ideation techniques** during the ideation session
- The system will shall provide **contextually relevant associations**
- The system shall be able to **trigger an artificial change** of direction during ideation session
- The system shall **suggest different conceptual views on the ideas, different refinements and transformations** of them as well as early product views

3. **Post Processing**

At the Post Processing phase context awareness functioning will have a collective role and not a supportive one. At the Post Processing phase the context awareness component shall collect the contextual associations created in the previous phases and save them as history data.
4 PROTOTYPE

For the purposes of this deliverable we created a prototype of the context awareness component of the idSpace platform. The prototype is based on the architecture described in Chapter 3, implemented in PHP. The aim of the first version of the prototype was to illustrate with programming examples how the context component will collect the context data, how this information will be presented to the users and set some rules which will be the starting point for creating an inference context engine for the idSpace platform.

The prototype manipulates the context data for the System and for Users. For the system, we use JavaScript for the collection of the client’s system information and an XML file containing this information is created. This file is uploaded to the server. The same file is also used as a log file tracking the user’s actions on the client side and by uploading it to the server the system is always aware of the user’s actions. Furthermore, we created an example of database tables containing users’ data as they defined by the use case scenarios. From the database scheme created we extracted an XML schema used for the implementation of the Validator. Based on the XML schema we created examples of XML data documents and we used them for the implementation and the demonstration of the context component prototype.

The implementation code of the prototype is given in Appendix B.

In this prototype, functions for the collection of the client’s system information were developed, as well as methods for uploading them to the server. A simulation of the Context Manager engine was also developed including the XML validation, the wrapper engine and a number of rules defined for the simulation of the Adaptation Manager. For the implementation, Ajax technology, JavaScript, PHP and XML technologies were used.
Definition and implementation of the conceptual model for Context Awareness in idSpace v1

Information for the logged user

Information for his system

Screen resolution: 1280*1024
Available view area: 1280*990
Color depth: 32
Buffer depth: undefined
DeviceDPI: undefined
LogicalDPI: undefined
FontSmoothingEnabled: undefined
PixelDepth: 32
UpdateInterval: undefined

Information for his browser

CodeName=Mozilla
MinorVersion=undefined
Name=Netscape
Version=5.0 (Windows; en-US)
CookiesEnabled=true
CPUClass=undefined
OnLine=true
Platform=Win32
UA=Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.5) Gecko/20080122 Firefox/3.0.5
BrowserLanguage=undefined
SystemLanguage=undefined
UserLanguage=undefined

List of Learners

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>george</td>
<td>learner</td>
</tr>
<tr>
<td>kostas</td>
<td>learner</td>
</tr>
<tr>
<td>George</td>
<td>Expert</td>
</tr>
</tbody>
</table>

Figure 13: Example of the Context Awareness Prototype

Figure 13 illustrates the client’s system information being collected, as implemented in the current prototype. It contains information like screen resolution and available view area. It also collects the browser information like the browser name, the version platform, etc. This information will form the System context element. On the same figure an example receiving recommendations is shown. The user is able to select the user role that he wishes to get recommendation for, or press a button asking for a recommendation. Figure 13 also shows the result of the recommendation for learners and experts.
5 CONCLUSION AND FUTURE WORK

The aim of this deliverable was to present the Architecture of the Context Awareness component of idSpace, as well as its first prototype implementation. An initial prototype of the context awareness component was implemented, taking into consideration the technologies used by the existing platform and the restrictions of the project.

The future work of Work Package 3, involves completing the Context Awareness component for the idSpace platform. Specifically, this can be divided in the following implementation and evaluation phases:

1. Collect relevant data for extracting context
2. Modeling of the context space
3. Adaptation reasoning and decision making
4. Evaluation of all the above
5. Improvements from evaluation feedback
REFERENCES


idSpace -2008-216199
Heider, T., (2008), idSpace D5.1 - idSpace User Requirements, http://hdl.handle.net/1820/1659


APPENDIX A

A.1 Functional Architecture

Context reasoning is referred to as the task of using context data in an intelligent way (Nurmi & Floréen, 2004). The reasoning for context aware applications has been approached from multiple perspectives and views. Through the research in reasoning problems, one of the main tasks was to define a functional architecture for the context reasoning procedures in context aware systems. The attempt to create systems which deduce the relevant information, from a bigger pool of context data led to the creation of a formal model of the context reasoning architecture.

The reasoning in context aware applications is performed by an individual engine which is adapted to a context aware system. This engine can have the form of a prototype, a middleware application or can be a component of a system.

To explain the functional architecture of a reasoning engine we will explain it as a single component within an overall architecture for context aware computing system. This is the same way (Padovitz, Loke, Zaslavsky, & Burg, 2004) are exploring a reasoning engine named “ReaGine” through a general overview of it. Padovitz et al., (2004) tracked five basic functional steps during the reasoning process taken by “ReaGine” as illustrated in Figure 14.

![Functional Architecture Diagram]

- Situation Composition
- Verification
- Conflict Analysis
- Knowledge Synthesis
- Low-Level Discrepancies Discovery
- Data Fusion
- Raw Data
Definition and implementation of the conceptual model for Context Awareness in idSpace v1

Figure 14: Reasoning Engine Functional Stages

Reasoning starts when information arrives to the reasoning engine either as raw data or as basic reasoned context, then the information is checked for low level discrepancies. At the next step the data is synthesized according to concepts drawn in the conceptual model, and then the engine checks for conflicts (e.g. ambiguous situations or situations that can not co-exist) at the conceptual level. Conflicting situations are dealt with in an additional verification phase for the resolution or the verification of the true situation’s nature. At the final stage the verified situations are composed of more complex situations and compared with policies predefined by the system designers.

In a more general view the reasoning engine receives the context data as raw data which is called ‘low level context’ and the engine transforms it to ‘higher level contexts’, which are combinations of lower level data sources (Nurmi & Floréen, 2004). The low level data are collected according to the actions or events coming from sensors or middleware software APIs defining the context data sources. The low level data are processed and mapped to high level context which is the data output of the engine.

Figure 15: Functional Architecture of Context Reasoning Engine

Based on the aforementioned, the Functional Architecture of Context Reasoning engine can be described as the combination of three sub-engines (Figure 15): the Event Collector engine, the Data Monitoring engine and the Rule or Adaptation engine.
The Event Collector engine is the engine which collects events and generates event notifications described according to an event model (Lee, Naganuma, & Kurakake, 2005). According to (Lee et al., 2005) event notification engines are using matching approaches like the matching based on simple flat, the hierarchical topic based event models or a graph structured event model. The simple flat matching and the hierarchical topic based event models are fast but they do not allow expressive subscriptions and sophisticated content based matching. Graph structured event model represents the different subscriptions as a direct acyclic graph in which the set of nodes in the graph is the set of possible subscription and notification categories (Kulik, 2003).

The Context Data Monitoring engines are related with the context inference engines that are described in the following sections. These engines use context reasoning techniques for the retrieval of context information.

The Adaptation Engine contains the functions and the formulas used by the system to make decisions for the adaptations results. The adaptation results are computed and extracted from the Adaptation Engine based on the user’s profile, history or any other parameter influencing the adaptation result.

A.2 Reasoning Techniques

In the literature there are numerous references for the role of context reasoning. Context reasoning, as it is described by the five functional stages of the reasoning engine, could be built as an ordered list composed of each event followed by any created reasoning events reason, including any predictions created by the engine as part of its process (Mastenbrook & Berkowitz, 2003). Thus context reasoning has a very important role as a part of the architecture of a context system. The context reasoning is responsible for the detection of possible errors, for the completion of missing values and for the decision of the quality and the validity of the sensed data. Its importance is focused on the responsibility of transformation of raw context data into meaningful information, and the extraction of decisions that may lead to actions.

In order to make the context reasoning tasks achievable, researchers deployed context reasoning techniques such as Ontological Reasoning, Rule Based Reasoning, Distributed Reasoning, Case Based Reasoning, Offline Reasoning and Probabilistic Reasoning. These techniques are separated and distinguished in Non-Symbolic reasoning Techniques and Symbolic Techniques which will be presented in the following sections. The classification of these techniques to symbolic or non-symbolic is depended on the methods of representation of the reasoning events and situations. The methods of representation might be deterministic by using symbols and regular expressions (Symbolic) or non deterministic using probabilistic methods (Non-Symbolic).
A.2.1 Non-symbolic Reasoning Techniques

Non-symbolic reasoning techniques use probabilistic methods and machine learning algorithms for the prediction of the proposed data. The architecture requirements of systems supporting learning algorithms for reasoning are not determined only by algorithms; they are also determined by data sets and the interaction of algorithms within a larger system (Crago et al., 2006). The architectural drivers based on the algorithm characteristics given by (Crago et al., 2006) are:

- **Probabilistic Relational Model**: Probabilistic Computation, large densely connected graphs, indirection over graph nodes, varying granularity, load balancing, trade off error for latency
- **SATisfiability – based planner**: Parallel tree traversal symbolic matching, partial results sharing and communication, any time solutions etc.
- **Support Vector Machine Classification**: Variable precision arithmetic on sparse vectors, flexible caching, computational density etc.

Non-Symbolic reasoning techniques are commonly used for the development of cognitive systems which are used to solve problems that are too difficult to be solved optimally or exactly so the use approximations and no zero error rates (Crago et al., 2006).

A.2.2 Symbolic Reasoning Techniques

Symbolic reasoning techniques are expressed with symbols. A very simple form of symbolic reasoning can be expressed as a single-case based simulation where each variable can have a single constant value only, and the association of an expression with each variable at each point of execution (Blank, Eveking, Levihn, & Ritter, 2001). Consider Figure 16 (Csallner, Tillmann, & Smaragdakis, 2008) as a simple example of a symbolic simulation.

```c
int test (int x, int y)
{
    int z=x*y;
    if (z<0)
    {
        //Throw exception
    }
    if (x<y)
    {
        int temp=x
        x=y;
        y=temp;
    }
}
```
When introducing symbolic simulation techniques, (Blank et al., 2001) mention that the symbolic expressions (like the example of Figure 16) presume a large number of cases for the provision of faster and more reliable results. In the same work they also give a number of the ingredients and problems of symbolic simulation:

1. the number of simulation steps is finite and limited
2. for the process of substitution the particular semantics of the function symbols involved do not care

Ontological Reasoning, Action-based Reasoning, Distributed Reasoning and Case-based Reasoning techniques can all be considered as Symbolic forms of reasoning.

An Ontological Reasoning example is CARE (Agostini, Bettini, & Riboni, 2006) CARE is a framework supporting online ontological reasoning for Context Aware Internet Services. The contextual data of the framework is managed by different entities (i.e. user, network operator, service provider). The context data is collected and managed by a certain entity and it is expressed by means of references to ontological classes and relations. The context management and reasoning is achieved by the declaration of policies in the form of rules over the collected data. Other examples of ontological reasoning are PELLET Anonymous, (2008a), and Racer Pro Anonymous, (2008b).

Action-based Reasoning

Action-based policies are undoubtedly the most popular and are used in different domains related to networks and distributed systems, such as computer networks, active databases, and expert systems. An action policy consists of situation-action rules, which specify exactly what to do in certain situations.

Fuzzy Rules

A possibility to introduce a more human-like way of thinking in rule-based reasoners is enabled by fuzzy logic (Wang, 1994) and fuzzy rules. Instead of having rules that are evaluated to either true or false (binary logic), fuzzy logic allows for multi-valued logic with different degrees of truth. Thus, instead of having IF-THEN-ELSE rules we have IF-THEN rules (without ELSE), or equivalent constructs, such as fuzzy associative matrices that allow multiple rules to be evaluated and assigned different degrees of truth for the same case. A typical fuzzy rule is defined in the following format: IF variable IS property THEN decision, where variables are linguistic rather than mathematical and properties take fuzzy rather than numerical values. Thus a fuzzy variable for temperature may take values such as very hot, hot and cold rather than a specific number in Celsius or Fahrenheit. The fuzziness of
properties along with the AND, OR and NOT operations of Boolean logic that also exist in fuzzy logic, allow as to assign scalar values to adaptation decisions depending on the antecedent of our rule.

**Goal-based Reasoning**

The benefit of goal-based policies for system administrators is that they are relieved from the task of defining actions (required in Action Based techniques), usually requiring a very detailed knowledge of the system. Under goal-based policies, the system itself must reason about a sequence of actions able to satisfy the goal.

**Utility Functions**

In the context of self-adaptive computing, one of the main needs is a way to detect the best variant among each possible state that the system is allowed to assume at a given time. Utility functions can be used in order to map each possible state of the system into a scalar value, so that it is possible to select automatically a new state for the system by selecting the configuration that provides the best utility value. They are mathematical equations that based on some input from the environment can assess the suitability of different adaptation alternatives.

**Case-based reasoning** is the process of solving problems based on the solutions of similar past problems (Anonymous, 2009a). The process is formalized as a four-step process:

1. **Retrieve**: Given a target problem, retrieve cases from memory relevant to solving it. A case consists of a problem, its solution, and, typically, annotations about how the solution was derived.

2. **Reuse**: Map the solution from the previous case to the target problem.

3. **Revise**: Having mapped the previous solution to the target situation, test the new solution in the real world (or a simulation) and, if necessary, revise.

4. **Retain**: After the solution has been successfully adapted to the target problem, store the resulting experience as a new case in memory.

Several Case Based reasoning applications were built, like *CadSyn* (Maher, 1996) which provides guidance for architectural design and adapts existing designs for new buildings. *AskJef* is an application using multimedia technology for storing and presenting cases to the user (Barber et al., 1992). Other Case Based Reasoning applications are *Casecad* (Maher, Balachandran, & Zhang, 1995) and *Archie-II* (Domeshek, Kolodner, & Zimring, 1994).

**Distributed reasoning** systems are the systems composed of separate modules (agents) and asset of communication paths between them. The separate modules might be individual reasoning sub paths implemented as Rule Based, Case Based or Ontological based reasoning systems. These modules can communicate between them and their combination is creating a...
distributed reasoning system. An example of Distributed reasoning system DRAGO (Serafini & Tamilin, 2005) a distributed reasoning Architecture for semantic web, which is implemented with multiple semantically related ontologies presented with different methods (Description Languages).

A.2.3 Hybrid Reasoning Techniques

Hybrid reasoning techniques combine two or more of the presented Symbolic or non-Symbolic Techniques. A very common combination of techniques is the integration of rule based and case based reasoning. The benefit of this combination is the creation of a new scheme which is derived from the individual schemes, e.g. rules derived from cases and vice versa (Prentzas & Hatzilygeroudis, 2002). Prentzas and Hatzilygeroudis (2002) support that the integration of hybrid reasoning techniques is distinguished in two categories: efficiency improving and accuracy improving methods. A hybrid approach is presented by (Prentzas & Hatzilygeroudis, 2002), but also by Gallant, (1993) and Ghalwash, (1998). Prentzas and Hatzilygeroudis (2002) proposed a hybrid reasoning approach integrating Symbolic rules with neuro-computing giving pre-eminence to the symbolic component. This approach is based on the example of hybrid approach given by Golding and Rosenbloom, (1996) which refers to the creation of an effective scheme combining three types of knowledge representation formalisms: symbolic rules, neural networks and cases.

Another hybrid reasoning approach was proposed by (Dejene, Scuturici, & Brunie, 2008). Their proposal is based on a relational model and an ontology-based approach for the creation of rules. The proposed hybrid context model, HCoM, suggests separating context data management from context knowledge management, process them separately and then put the results together for better reasoning and decision support in a context aware environment.

A.3 Technologies for Context Reasoning

The various context reasoning techniques that were presented in the previous section can be implemented using a number of different technologies. Here we visit the ones that are most relevant for the idSpace case:

- The use of rule based reasoning engine technologies for context reasoning
- Ontology based reasoning technologies
- Use of the Topic Maps technology

A.3.1 Context abstraction using rule-based reasoning engines

Rule-based reasoning is performed through the conversion of context data into “facts” and the creation of rules to obtain new facts to be converted back as new context information. The functioning of a rule-based reasoning engine can be summarized as follows:
1. A Context Reasoning Provider is used for the creation of specific context reasoning sessions and their activation on specific context reasoning situations. It manages “WHEN” a context session is created.

2. Context Abstraction Rules are created to capture “HOW” the abstract context information collected by the Context Reasoning Provider is used to make the context data information more specific.

3. The context information is translated in facts to represent knowledge and to activate context abstraction rules.

4. The new rules derived from the context abstraction rules are converted back to the context representation and made available as new context information.

Taveter and Wagner, (2001) in an attempt to give a definition for the Business Rules they specified the types of Business Rules that exist in the literature. These types are extracted and applied to a model and formalize the types of rules. The types mentioned by (Taveter & Wagner, 2001) can be used to define the types of rules used for the definition of an abstract rule based reasoning engine. The types of rules are:

- **Integrity constraints**: Integrity constraint rule is an assertion that is based on the evolved states and the transition histories of a system viewed as a discrete dynamic system (Wagner, 2002). Constraints can be expressed as IF-THEN statements in programming languages or as explicit assertion statements supported by programming languages such as C++, or Java 2.

- **Derivation rules**: Derivation rules allow the derivation of knowledge from other knowledge by an inference or a mathematical equation (Wagner, 2002).

- **Reaction rules**: The Reaction Rules define the behavior of a system by stating the conditions under which actions must be taken. These conditions are expressed in response to perceived environment events and communication events (Wagner, 2002).

Wagner (2002) also gives examples of the methods of modeling and representing the rules with UML (Unified Modeling Language), SQL database queries and XML based languages. For the implementation of the rule based reasoning, several rule based engines and technologies are available, like Mandarax (Dietrich, 2003), ILOG (Anonymous, 2009a) and Jess (Anonymous, 2008d).

**Mandarax** (Dietrich, 2003), is an open source java library for business rules, including the representation, persistence, exchange, management and processing of rule bases. In Mandarax rules are presented as clauses. The clauses consist of a body which is the prerequisite of a rule and a head which is the consequence of a rule. Both of the clauses are facts which themselves consist of terms and predicates associating those terms. Terms can be constants, variables or complex terms.

**ILOG** (Anonymous, 2009a) is a rule engine and programming library used for the combination of rule based and object oriented programming for the adjustment of business rules to
Definition and implementation of the conceptual model for Context Awareness in idSpace v1

new and existing applications. ILOG rule is composed of a header, a condition part and an action part. The header defines the name of the rule, its priority and packet name. The condition part of the rule defines the conditions that must be met such that the rule is eligible for execution. The action part specifies the activities to be carried out when the rule is fired. The ILOG engine can directly parse and output XML representation allowing the management of rules by standard XML tools.

Jess (Anonymous, 2008d) is one of the most popular rule engines. It is a Java library which offers different levels of APIs for the creation of reasoning sessions, load facts and rules and run reasoning algorithms. Jess implements the RETE reasoning algorithm to fire rules and derive new facts.

A.3.2 Ontology-based inference engines

In this subsection a number of ontology –based inference engines will be described. The described engines will be F-OWL, OWL inference engine based on XSLT and JESS, Jena 2, SWRL rules from Protégé, BaseVIisor, RacerPro, FaCT++, Pellet and KAON2.

F-OWL is an ontology inference engine for the Web Ontology Language OWL. The engine is using the Flora2 advanced object oriented knowledge base language which translates a unified language of F-Logic, HiLog and translation Logic into the XSB deductive engine. F-OWL is reasoning using the ontology model defined by the standard OWL language. F-OWL has been used as ontology reasoner in a number of intelligent prototypes like CoBra, TAGA and REI.

OWL inference engine based on XSLT and JESS is an inference engine which is using the JESS rule engine and XSLT style sheets. JESS has the ability to “reason” using knowledge supplied in the form of declarative rules, and for the translation to Jess rules using the JESS for OWL some kind of “adapters” are needed.

The code provided by the project helps to load OWL ontologies and annotations into JESS, transforming them into rules and facts. Then using CLIPS (syntax) inference rules for the ontology are created. The code is organized as follows:

**• OWL Meta model: description of OWL meta-model in JESS language and directly loaded into the JESS engine.**

**• Ontology Stylesheet: an XSLT style sheet that transforms an OWL schema into a set of JESS assertions based on the OWL Meta model. The resulted assertions can be loaded into the JESS engine.**

**• Annotation Stylesheet: an XSLT style sheet which transforms an OWL annotations file into a set of JESS assertions based on OWL meta-model. The resulted assertions can be loaded into the JESS engine.**

Jena 2 is an inference subsystem which allows to a range of inference engines or reasoners to be plugged into Jena. The engines are used to derive additional RDF assertions which are
entailed from some base RDF together with any optional ontology information and the axioms and rules associated with the reasoner. With this mechanism languages like RDFS and OWL can be supported. With the support of these languages the additional facts to be inferred from instance data and class descriptions is allowed. Jena2 includes the predefined reasoners for RDFS, OWL-Lite, DAML and other generic reasoners for user defined rules.

**SWRL rules from Protégé** is an inference engine with a more complete solution related with Semantic Web. SWRL Bridge which provides the necessary the infrastructure to incorporate rule engines into Protégé – OWL for the execution of SWRL rules, is an add-in to the engine. The interaction with this bridge is achieved through a user interface called SWRLJessTab.

**BaseVIsor** is a forward-chaining inference engine based on the Rete network optimized for the processing of RDF triples. It has outfitted to process RuleML and R-Entailment rules. It is a Rule Based system implementing a Rete based algorithm solution. The difference with JESS and CLIPS is the fact that the user defined types cannot be arbitrary list structures but simple data structures in order to have increased efficiency of rules pattern matching.

**RacerPro** is a core inference engine for the semantic web. It has two APIs that are used by network clients like OilEd, the visualization tools RICE and the ontology development environment Protégé 2. The Racer server supports the standard DIG protocol via HTTP and a TCP based protocol with extensive query facilities. It supports the web ontology languages DAML+OIL, RDF and OWL.

**FaCT++** is a description logic reasoner written in C++. It can be used with applications supporting OWL-DL via the DIG standard; it uses a tableaux decision procedure to solve SHOIQ description logic and supports simple data types.

### A.3.3 Topic Map Technologies

Topic Maps technology is part of the ISO standards of the semantic web technologies. The ISO standard of Topic Maps is formally known as ISO 13250. In general, topic maps are used in semantic web applications for finding and exchanging information using topics. Topic Maps are designed for the enhancement of navigation and information retrieval. This is achieved by the addition of semantics into the resources and their representation as context data sets (Hatzigaidas, Papastergiou, Tryfon, & Maritsa, 2004).

Wrightson, (2001) through the description of Topic Maps mentions the distinguished differences of a topic map and ontologies. Ontologies are used for the description of shared common understanding aspects, like objects or relations between them. A topic map is used for the representation of the ontologies by linking the resources belonging to them. In other words a topic map is the collection of topics, associations and scopes (Anonymous, 2001). The consisting parts of a topic as they are introduced by (Anonymous, 2001) are the subject which gives the generic sense of a topic, the reification which is the act for the creation of a topic, the subject identity which is used to specify the relation between subjects, the subject
**indicator** which is a resource indicated by the topic and finally the topic characteristics which are the name, the role the occurrence or the associations of a topic.

Topic Maps technology is used for the integration of several Topic Maps standards. Topic Maps APIs, Query standards, Constraint standards (Anonymous, 2008a), are:

- TMAPI (Topic Maps Application Programming Interface)
- ISO:1848 TMQL (Topics Map Query Language)
- ISO 19756: TMCL (Topic Maps Constraint Language)
- XTM (XML Topic Maps)

In the (Hatzigaidas et al., 2004) research work three Topic Maps Tools categories were identified: *Topic Maps Engines, Topic Maps Navigators* and *Topic Maps Editors*.

**Topic Maps Engines**

**Omnigator** (Anonymous, 2007): Omnigator is a technology showcase and teaching used as a Topic Map debugger and prototyping tool. It can debug and represent everything is provided as topic map and it lets the user load and navigate any conforming topic map whether its format is XTM, HyTM, LTM or RDF. It allows the navigation of topic maps in a generic interface and the quick test of the topic maps. The common use of the engine has educational role because it is a free web based application used for the better understanding of Topic Maps by building and debugging demo applications.

**SemanText** (Anonymous, 2008f): SemanText is a prototype application which is used to demonstrate how the topic map standard can be used for the representation of semantic networks. It builds a knowledge base in the form of semantic network from a topic map and it is commonly used for the creation of inference engines and expert systems.

**Tm4Jscript** (Anonymous, 2003b): Tm4Jscript is a topic map engine entirely written and developed with Java scripts. It is based on the reverse engineering of xtm into an object model. For a better performance it uses indexes and hash tables and it uses a number of methods to avoid query languages. It includes JTMA (Java script Topic Maps Application) and it can also be used as a browser application.

**Tmproc** (Anonymous, 2008g): tmproc engine is a Python implementation of the 13250 Standard. It is used for the importation, exportation of queries and Topic Maps manipulation. It requires the PyXML (Anonymous, 2003b) package and can also be used with xmlarch (Anonymous, 2009d) if architectural design is needed.

Other Topic Maps Engines are:

- empolis k42 (http://xml.coverpages.org/empolisK42Ann.html),
- TM4J (http://tm4j.org/),
• and xSiteable (http://xsiteable.sourceforge.net/).

**Topic Maps Navigators**

**HyperGraph** (Anonymous, 2003a): Hypergraph is used for the visualization of Topic Maps. It is an open source project which provides java code to work with hyperbolic geometry and hyperbolic trees. It provides a very extensible API to visualize hyperbolic geometry, to handle graphs and to layout hyperbolic trees.

**Panckouke** (Anonymous, 2004): Panckouke is a java library which supports the development of topic map presentation and navigation tools. It is a subproject of the tm4j project and it is used as the underlying topic map engine of it. It supports the automatic creation of topic maps from structured data.

**The “V” Topic Map Browser**: V browser is a prototype of an HTML topic map browser, written in Python, having the Gooseworks toolkit as its backend.

**TM3D**: TM3D is used for the visualization of topic maps in 3D. TM3D is a graphical tool allowing the 3D navigation of XML Topic Maps. It uses OpenGL and Xerces and it can be used just as a viewing tool.

**TMNav**: A subproject of TM4J for browsing Topic Maps. It can connect to any of the backends of TM4J and browse Topic Maps. It has two methods of navigation. The standard GUI and a dynamic graph GUI which is using the TouchGraph Library. With the use of tolog 1.0, the merging and creation of Topic Maps is also supported.

Other Topic Maps navigators are:

• TM4Web (http://tm4j.org/tm4web.html)

• ThinkGraph (http://www.thinkgraph.com/english/index.htm)

**Topic Maps Editors**

Some of the existing Topic Maps editors are:

• Mapalizer a Perl developed tool for rapid topic map authoring

• Topic Map Designer, an editor and graph viewer all in one.

• TMTab a tab-widget plug-in for Protégé. It uses the Protégé as a topic map editing tool. It supports the export of XTM files from an ontology developed in Protégé which is based in the “topic map ontology”. It can use the Protégé slots; it creates classes for the representation of topic names, occurrences and the establishment of associations between topics.
A.4 Technologies for Context Storage and Retrieval

Advances in context awareness research made the creation and development of context storage and retrieval, mandatory. The need for adaptation methods and the generation of recommendations within a computing system indicated that such a system is context aware. The dynamic change of data, on runtime, the storage and retrieval of data guided the research world to find solutions based on the context technologies they integrated. (Siljee, Bosloper, & Nijhuis, 2004) divide the concept of context awareness in three phases:

1. Monitoring of the environment,
2. Interpretation of the data (being monitored) through the context model and
3. Adaptation of the system to the changed context.

These three phases predicated upon the interaction with a repository of the context data and the integration of mechanisms for the update, storage and retrieval of them.

In a more detailed analysis, (Siljee et al., 2004) are tracking the different aspects in the process of retrieving the context information. These aspects are identified as initiative, timing, history, and information presentation.

Initiative is the process of context information retrieval using context-push and context-pull. With context-push the context information is retrieved without the need of send request to the system every time the information is needed. With context-pull, when the system needs context information it explicitly request for the needed information.

Timing includes the event driven and periodic methods of context information retrieval. Event driven method requests for context information when some event occurs. Periodic method requests or retrieves context information in certain scheduled points in time.

History includes the absolute and relative methods. Absolute method context information is taken in absolute points without using previous information. Relative is based on the difference of context information over time, thus the previous stored information is used.

Information presentation is divided in explicit and implicit. The explicit method does not need any context model and it uses the needed information as it is. With implicit method the system refers to the context information based on its input data and thus a context model is needed.

In the following section some of the existing technologies for storage and retrieval of context data will be presented and based on them, conclusions for the context storage and retrieval technologies in idSpace will be extracted.

A.4.1 Existing technologies for context storage and retrieval

Existing technologies for context storage and retrieval use the classical RDBMS (Relational Data Base Managing System) for the storage of data. The classical RDBMS includes the MySQL, (Anonymous, 2008e) PostgreSQL, Oracle, MSSQL server and other relational database engines. These engines are commonly used as the repository engines for data and they
have more specific role within the context aware applications. Semantic web tools like Jena, Protégé, and others support the storage of RDF (Resource Description Framework) triples and OWL ontologies. The majority of the existing solutions of context storage and retrieval are using RDF storage systems. Some of them are described in this subsection.

3store

Harris and Gibbins, (2003) introduced the 3store platform. 3store is a C core library which is using MySQL for storing its raw RDF data and caches. Initially it was developed for POSIX compliant UNIX systems using ANSI C. The efficiency of C language was giving the flexibility to interface it in other languages like PHP, Java, PYTHON, Perl and C++.

The 3store engine is represented as a 3 layer model, RDF Syntax, RDF representation and Relational Database System (Harris & Gibbins, 2003). The platform is using an off the shelf RDF parser and passes the resulted triples 3store’s rdfsql library to assert them into the knowledge base. This library translates the triples into SQL INSERT statements and passes them to the RDBMS. The library is also offering OKBC and RDQL query interfaces over HTTP, or directly through the C library.

Redland

Redland is a set of free software libraries that provide support for the RDF. It is a set of modular object based libraries APIs for manipulation graphs triples URIs and Literals. Redland appears to be capable of storage of large RDF graphs in memory and persistently with Sleepycat/Berkeley DB, MySql, PostgreSQL, AKT, Triplestore, SQLite, files or URIs. It supports multiple syntaxes for reading and writing RDF as RDF/XML, N-Triples and Turtle Terse RDF Triple Language, RSS etc. It creates SPARQL and RDQL queries and uses the Raptor RDF Parser Library. It can be bind with other languages like Perl, PHP, and Python via the Redland Bindings package.

Joseki/Jena

Joseki is part of the Jena RDF framework. It is an HTTP engine that supports the SPARQL Protocol and the SPARQL RDF Query language. It can manipulate RDF data from files and database storage engines. It uses GET and POST methods for the HTTP implementation of the SPARQL protocol.

Sesame

Sesame is a Java based open source RDF framework and repository with support for RDF Schema inference and querying. Sesame is supported by SAIL (Storage and Inference Layer) API that offers specific methods to its clients and translates these methods to calls to its specific DBMS. Thus Sesame is able to be implemented on the top of a wide variety of repositories without changing any of its components (Broekstra, Kampman & Harmelen, 2002). It has a flexible method of selecting the communication method of its modules. For example it may prefer communication over HTTP for a web context and for other web contexts protocols like Remote Method Invocation (RMI) or Simple Object Access Protocol (SOAP) may be more suited (Broekstra et al., 2002). Plug-ins exist that enable Sesame to be used also as an RDF repository from the Jena library.
**OWLIM Semantic Repository**

**OWLIM** is a high performance semantic repository, packaged as SAIL for the Sesame RDF database. It is based on TRREE a native rule entailment engine. OWLIM is available in two versions, **SwiftOWLIM** and **BigOWLIM**. SwiftOWLIM performs reasoning and query evaluation in-memory, while a reliable persistence strategy assures data preservation, consistency, and integrity. SwiftOWLIM is the fastest RDF(S) and OWL engine. BigOWLIM operates directly with binary persistence files, which allows it to scale to billions of statements. BigOWLIM is the only engine proven to support non-trivial OWL inference against 3 Billion triples (Anonymous, 2009b).

**AllegroGraph 64-bit RDF Store**

AlegroGraph is a high performance, high volume RDF storage for very efficient loading, indexing, storage and querying of millions of RDF triples, and has been also designed with a special focus on robustness. Alegro provides APIs for different languages, such as Java, Lisp or Prolog. It is best suited for 64-bit operating systems.
APPENDIX B

B.1 Java Script function for Browsing with AJAX technology

```javascript
var xmlHttp

function RecommendUsers(str)
{
  xmlHttp=GetXmlHttpObject();
  if (xmlHttp==null)
  {
    alert ("Your browser does not support AJAX!");
    return;
  }
  var url="getRecommendedUsers.php";
  url=url+"?q="+str;
  url=url+"&sid="+Math.random();
  xmlHttp.onreadystatechange=stateChanged;
  xmlHttp.open("GET",url,true);
  xmlHttp.send(null);
}

function stateChanged()
{
  if (xmlHttp.readyState==4)
  {
    document.getElementById("txtRecs").innerHTML=xmlHttp.responseText;
  }
}

function GetXmlHttpObject()
{
  var xmlHttp=null;
  try
  {
    // Firefox, Opera 8.0+, Safari
    xmlHttp=new XMLHttpRequest();
  }
  catch (e)
  {
    // Internet Explorer
  }
}
```
try
{
    xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
}
catch (e)
{
    xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
}
return xmlHttp;

B.2 Java Script Functions for collecting System Information from the Client

function GetUrl(){
    return location.href;
}
function ScreenInfo(){
    document.write("Screen resolution: ");
    document.write(screen.width + "*" + screen.height);
    document.write("<br />");
    document.write("Available view area: ");
    document.write(screen.availWidth + "*" + screen.availHeight);
    document.write("<br />");
    document.write("Color depth: ");
    document.write(screen.colorDepth);
    document.write("<br />");
    document.write("Buffer depth: ");
    document.write(screen.bufferDepth);
    document.write("<br />");
    document.write("DeviceXDPI: ");
    document.write(screen.deviceXDPI);
    document.write("<br />");
    document.write("DeviceYDPI: ");
    document.write(screen.deviceYDPI);
    document.write("<br />");
    document.write("LogicalXDPI: ");
}
Definition and implementation of the conceptual model for Context Awareness in idSpace v1

document.write(screen.logicalXDPI);
document.write("<br />");
document.write("LogicalYDPI: ");
document.write(screen.logicalYDPI);
document.write("<br />");
document.write("FontSmoothingEnabled: ");
document.write(screen.fontSmoothingEnabled);
document.write("<br />");
document.write("PixelDepth: ");
document.write(screen.pixelDepth);
document.write("<br />");
document.write("UpdateInterval: ");
document.write(screen.updateInterval);
document.write("<br />");
}

function BrowserInfo(){
    var x = navigator;
    document.write("CodeName=" + x.appCodeName);
    document.write("<br />");
    document.write("MinorVersion=" + x.appMinorVersion);
    document.write("<br />");
    document.write("Name=" + x.appName);
    document.write("<br />");
    document.write("Version=" + x.appVersion);
    document.write("<br />");
    document.write("CookieEnabled=" + x.cookieEnabled);
    document.write("<br />");
    document.write("CPUClass=" + x.cpuClass);
    document.write("<br />");
    document.write("OnLine=" + x.onLine);
    document.write("<br />");
    document.write("Platform=" + x.platform);
    document.write("<br />");
    document.write("UA=" + x.userAgent);
    document.write("<br />");
    document.write("BrowserLanguage=" + x.browserLanguage);
    document.write("<br />");
    document.write("SystemLanguage=" + x.systemLanguage);
}
B.3 Java Script for the creation of XML file with the System’s data

```javascript
function saveXML() //Creates XML File
{
  var TristateFalse = 0;
  var ForWriting = 2;
  myActiveXObject = new ActiveXObject("Scripting.FileSystemObject");
  myActiveXObject.CreateTextFile("C:\wamp\www\idSpace\ca\file.xml");
  file = myActiveXObject.GetFile("C:\wamp\www\idSpace\ca\file.xml");
  text = file.OpenAsTextStream(ForWriting, TristateFalse);
  var stringText="<?xml version=’1.0’ encoding=’utf-8’?>
  stringText+="<SystemInfo>
  stringText+="<urlLocation>"+location.href+"</urlLocation>
  stringText+="<ScreenInfo>
  stringText+="<resolution>"+screen.width + "*" + screen.height+"</resolution>
  stringText+="<viewArea>"+screen.availWidth + "*" + screen.availHeight+"</viewArea>
  stringText+="</ScreenInfo>
  stringText+="<BrowserInfo>
  stringText+="<browsername>"+navigator.appCodeName+"</browsername>
  stringText+="<version>"+navigator.appMinorVersion+"</version>
  stringText+="</BrowserInfo>
  stringText+="</SystemInfo>
  text.Write(stringText);
  text.Close();
}
```

B.4 Example of the created XML file with the System’s data

```xml
<?xml version='1.0' encoding='utf-8'?>
<SystemInfo>
<urlLocation>http://localhost/idSpace/ca/test1.php</urlLocation>
<ScreenInfo>
<resolution>1280*1024</resolution>
&viewArea>1280*990</viewArea>
</ScreenInfo>
<BrowserInfo>
```
B.5 XML Schema for idSpace example

```xml
<?xml version="1.0" encoding="utf-8"?>
  <xs:annotation>
    <xs:appinfo source="urn:schemas-microsoft-com:xml-msdatasource">
      <DataSource DefaultConnectionIndex="0" FunctionsComponentName="QueriesTableAdapter" Modifier="AutoLayout, AnsiClass, Class, Public" SchemaSerializationMode="IncludeSchema" xmlns="urn:schemas-microsoft-com:xml-msdatasource">
        <Connections>
            <Tables>
              <TableAdapter BaseClass="System.ComponentModel.Component" DataAccessorModifier="AutoLayout, AnsiClass, Class, Public" DataAccessorName="DataTable1TableAdapter" GeneratorDataComponentClassName="DataTable1TableAdapter" Name="DataTable1" UserDataSetName="DataTable1TableAdapter">
                <MainSource>
                  <DbSource ConnectionRef="TestForIdSpaceConnectionString (Web.config)" DbObjectType="Unknown" FillMethodModifier="Public" FillMethodName="Fill" GenerateMethodAndProperty="Both" GenerateShortCommand="false" GenerateGetMethodName="CA" GeneratorSourceName="Fill" GetMethodModifier="Public" GetMethodName="CA" QueryType="Rowset" ScalarCallReturnValue="System.Object, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" UseOptimisticConcurrency="false" UserGetMethodName="CA" UserSourceName="Fill">
                    <SelectCommand>
                      <CommandText>SELECT users.userId, users.userName, users.userSurname, users.userRole, users.usrInterests, portofolio.ideaId, portofolio.userId AS portofolioUserId, ideas.ideaId AS ideaId, ideas.ideaDescription, tasks.taskId, tasks.taskDescription, tasks.taskDescription AS taskDescription, tasks.Objective AS objective, tasks.domainId AS taskDomainId, roles.roleId, roles.roleDescription, domains.domainId AS domainId, domains.domainDescription FROM users INNER JOIN portofolio.usId AS portofolio.userId ON users.userId = portofolio.userId INNER JOIN ideas ON portofolio.ideaId = ideas.ideaId INNER JOIN users.userId = portofolio.ideaId CROSS JOIN tasks INNER JOIN roles.roleId ON tasks.roleId = roles.roleId INNER JOIN domains.domainId ON tasks.domainId = domains.domainId</CommandText>
                      <XmlCriteria/>
                    </SelectCommand>
                </MainSource>
              </TableAdapter>
            </Tables>
          </Connections>
        </DataSource>
      </xs:appinfo>
    </xs:annotation>
  </xs:schema>
</SystemInfo>
</BrowserInfo>
</version>0</version>
</SystemInfo>
"Mozilla</browsername>"
domains ON tasks.domainId = domains.domainId CROSS JOIN roles

</CommandText>
</DbCommand>
</SelectCommand>
</DbSource>
</MainSource>
</Mappings>
</Sources>
</TableAdapter>
</Tables>
</Sources>
</xs:appinfo>
</xs:annotation>

<xs:element name="CA" msdata:IsDataSet="true" msdata:UseCurrentLocale="true" msprop:Generator_UserDSName="DataSet 1" msprop:Generator_DataSetName="DataSet1">

</xs:complexType>
</xs:element>
Definition and implementation of the conceptual model for Context Awareness in idSpace v1
Definition and implementation of the conceptual model for Context Awareness in idSpace v1
Definition and implementation of the conceptual model for Context Awareness in idSpace v1

```xml
<xs:complexType>
  <xs:sequence>
    <xs:element name="domainDescription" msprop:Generator_ColumnPropNameInRow="domainDescriptionColumn"
                msprop:Generator_ColumnVarNameInTable="columndomainDescription"
                msprop:Generator_ColumnPropNameInTable="domainDescreptionColumn">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:maxLength value="2147483647"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="portofolioUserId" msprop:Generator_UserColumnName="portofolioUserId"
                msprop:Generator_ColumnPropNameInRow="portofolioUserIdColumn"
                msprop:Generator_ColumnVarNameInTable="columnportofolioUserId"
                type="xs:int" minOccurs="0"/>
    <xs:element name="objective1" msprop:Generator_UserColumnName="objective1"
                msprop:Generator_ColumnPropNameInRow="objective1Column"
                msprop:Generator_ColumnVarNameInTable="columnobjective1"
                type="xs:base64Binary" minOccurs="0"/>
    <xs:element name="taskDomainId" msprop:Generator_UserColumnName="taskDomainId"
                msprop:Generator_ColumnPropNameInRow="taskDomainIdColumn"
                msprop:Generator_ColumnVarNameInTable="columntaskDomainId"
                type="xs:int"/>
    <xs:element name="dimainId" msdata:ReadOnly="true" msdata:AutoIncrement="true"
                msdata:AutoIncrementSeed="-1"
                msdata:AutoIncrementStep="-1"
                msprop:Generator_UserColumnName="dimainId"
                msprop:Generator_ColumnPropNameInRow="dimainIdColumn"
                msprop:Generator_ColumnVarNameInTable="columndimainId"
                type="xs:int"/>
    <xs:element name="ideaId1" msdata:ReadOnly="true" msdata:AutoIncrement="true"
                msdata:AutoIncrementSeed="-1"
                msdata:AutoIncrementStep="-1"
                msprop:Generator_UserColumnName="ideaId1"
                msprop:Generator_ColumnPropNameInRow="ideaId1Column"
                msprop:Generator_ColumnVarNameInTable="columnideaId1"
                type="xs:int"/>
    <xs:element name="taskId1" msdata:ReadOnly="true" msdata:AutoIncrement="true"
                msdata:AutoIncrementSeed="-1"
                msdata:AutoIncrementStep="-1"
                msprop:Generator_UserColumnName="taskId1"
                msprop:Generator_ColumnPropNameInRow="taskId1Column"
                msprop:Generator_ColumnVarNameInTable="columntaskId1"
                type="xs:int"/>
    <xs:element name="taskDescription" msprop:Generator_UserColumnName="taskDescription"
                msprop:Generator_ColumnPropNameInRow="taskDescriptionColumn"
                msprop:Generator_ColumnVarNameInTable="columntaskDescription"
                minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

idSpace -2008-216199
B.6 XML Context Data Example

```xml
<p1:CAdata>
    <p1:userId>1</p1:userId>
    <p1:userName>George</p1:userName>
    <p1:userSurname>Sielis</p1:userSurname>
    <p1:userRole>Learner</p1:userRole>
    <p1:usrInterests>HCI, E-Learning</p1:usrInterests>
    <p1:ideaId>1</p1:ideaId>
    <p1:ideaDescription>…</p1:ideaDescription>
    <p1:taskId>2</p1:taskId>
    <p1:taskDescription>……</p1:taskDescription>
    <p1:Objective>…</p1:Objective>
    <p1:domainId>-3</p1:domainId>
    <p1:roleId>4</p1:roleId>
    <p1:roleDescription>…..</p1:roleDescription>
    <p1:domainDescription>Computer Science</p1:domainDescription>
    <p1:portfolioUserId>……..</p1:portfolioUserId>
    <p1:objective1>…….</p1:objective1>
    <p1:taskDomainId>147483648</p1:taskDomainId>
    <p1:domainId>-7483648</p1:domainId>
    <p1:ideaId1>147483648</p1:ideaId1>
    <p1:taskId1>147483648</p1:taskId1>
    <p1:taskDescription>……</p1:taskDescription>
</p1:CAdata>
```
B.7 XML Validator in PHP

```php
<?php
function libxml_display_error($error)
{
    $return = "<br/>
";
    switch ($error->level) {
        case LIBXML_ERR_WARNING:
            $return .= "<b>Warning $error->code</b>: ";
            break;
        case LIBXML_ERR_ERROR:
            $return .= "<b>Error $error->code</b>: ";
            break;
        case LIBXML_ERR_FATAL:
            $return .= "<b>Fatal Error $error->code</b>: ";
            break;
    }
    $return .= trim($error->message);
    if ($error->file) {
        $return .= " in <b>$error->file</b>";
    }
    $return .= " on line <b>$error->line</b>\n";

    return $return;
}

function libxml_display_errors() {
    $errors = libxml_get_errors();
    foreach ($errors as $error) {
        print libxml_display_error($error);
    }
    libxml_clear_errors();
}

// Enable user error handling
libxml_use_internal_errors(true);
```
$xml = new DOMDocument();
$xml->load('ca2.xml');

if (!$xml->schemaValidate('DataSet1.xsd')) {
    print '<b>DOMDocument::schemaValidate() Generated Errors!</b>
    libxml_display_errors();
} else {
    echo "validated";
    $xmlValid=true;
}

?>

B.8 PHP reading / writing XML data files

<?php
    $xml = new DOMDocument();
    $xml->load('ca1.xml');

    $xsl = new DOMDocument();
    $xsl->load('ca1.xsl');

    $proc = new XSLTProcessor;
    $proc->importStyleSheet($xsl);

    echo $proc->transformToXML($xml);

?>