THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Université Toulouse III Paul Sabatier*
Discipline ou spécialité : *Informatique*

Présentée et soutenue par *Valentin BUTOIANU*
Le *04 Avril 2013*

Titre :
Share and Reuse of Context Metadata Resulting from Interactions between Users and Heterogeneous Web-based Learning Environments

---

JURY
M. Thierry NODENOT, Professeur à l'Université de Pau, Président
M. Serge GARLATTI, Professeur à TELECOM Bretagne, Brest, Rapporteur
M. Alain MILLE, Professeur à l'Université Claude Bernard Lyon I, Rapporteur
M. Christophe REFFAY, MCF à l'ENS de Cachan, Examinateur
M. Julien BROISIN, MCF à l'Université Toulouse III Paul Sabatier, Encadrant

---

Ecole doctorale : *Mathématiques, Informatique, Télécommunications de Toulouse (MITT)*
Unité de recherche : *Institut de Recherche en Informatique de Toulouse CNRS - UMR - 5505*
Directeur de Thèse : *M. Philippe VIDAL, Professeur à l'Université Toulouse III Paul Sabatier*

... to my grandparents, Victor and Paveluţa Smaranda.

... pentru bunicii mei, Victor şi Paveluţa Smaranda.

Acknowledgements

First of all I would like to express my gratitude to Mr. Julien Broisin, Associate Professor at Paul Sabatier University of Toulouse. His constant patience, encouragements, support and invaluable suggestions made this thesis work successful. His ability to answer questions in a very precise way has always been of great help. I have been very lucky to have a supervisor who cared so much about my work.

My sincere thanks go to Mr. Philippe Vidal, Professor at the Paul Sabatier University of Toulouse, for giving me the opportunity to make this thesis, for his confidence, his everyday kindness and his skill to promptly respond to all my requirements of various natures during all these years.

I would also like to thank the Human Computer Interaction Education (HCI) group of the Computer Science Department of Catholic University of Leuven, Belgium. The two months spent there, played an important role in my work. I want to thank Mr. Erik Duval, the head of the group, for his good advice, Mrs. Katrien Verbert for the fruitful collaboration we had, and Mr. Gonzalo Parra, for his help as a colleague and a friend.

Completing this work would have been all more difficult without the support and friendship provided by the other members of the SIERA research team, led by Mr. Abdelmalek Benzekri. I am indebted to them for their help, everyday sympathy and good humor.

I must express my gratitude to Mădălina Mitran, my girlfriend, for her continuous support and encouragements. She was always near me, she understood me with patience, and gave me her good advice.

I would like to thank my best friend Dr. Petar Lukanov with whom I spent the best crazy moments of my life in Toulouse and I am sure we will remain friends forever.

I also thank my friends Dana Codreanu, Adrian Chifu, Andra Codreanu, Rémi Venant, Cristina Stoican, Ana-Maria Mânzat and Anca Alimănescu for providing the comprehension, support and friendship that I needed during the good as well as difficult moments. I have also to thank my roommates Tetiana Krachko and Mykhailo Ianchuk, for their everyday gentleness. The very good moments we spent together, I will always remember.

I want to thank also to Mrs. Nadia Yasinne-Diab, English teacher at Paul Sabatier University of Toulouse, for her kindness to accept to read this document in order to improve its quality

in terms of English grammar.

Finally, but most importantly, I would like to thank my family: my parents, Ion and Maria Butoianu and my sister, Elena Croitoru for their constant support. Without their everyday encouragements it would have been much harder for me to finish this work.

Abstract

An interest for the observation, instrumentation, and evaluation of online educational systems has become more and more important within the Technology Enhanced Learning community in the last few years. Conception and development of Adaptive Web-based Learning Environments (AdWLE) in order to facilitate the process of re-engineering, to help understand users' behavior, or to support the creation of Intelligent Tutoring Systems represent a major concern today. These systems handle their adaptation process on the basis of detailed information reflecting the context in which students evolve while learning: consulted resources, mouse clicks, chat messages, forum discussions, visited URLs, quizzes selections, and so on.

The works presented in this document are intended to overcome some issues of the actual systems by providing a privacy-enabled framework dedicated to the collect, share and reuse of context represented at two abstraction levels: raw context (resulting from direct interactions between users and applications) and inferred context (calculated on the basis of raw context). The framework is based on an open standard dedicated to system, network and application management, where the context specific to heterogeneous tools is represented as a unified and extensible structure and stored into a central repository. To facilitate access to this context repository, we introduced a middleware layer composed of a set of tools. Some of them allow users and applications to define, collect, share and search for the context data they are interested in, while others are dedicated to the design, calculation and delivery of inferred context.

To validate our approach, an implementation of the suggested framework manages context data provided by three systems: two Moodle servers (one running at the Paul Sabatier University of Toulouse, and the other one hosting the CONTINT project funded by the French National Research Agency) and a local instantiation of the Ariadne Finder. Based on the collected context, relevant indicators have been calculated for each one of these environments. Furthermore, two applications which reuse the encapsulated context have been developed on top of the framework: a personalized system for recommending learning objects to students, and a visualization application which uses multi-touch technologies to facilitate the navigation among collected context entities.

Keywords: Adaptive Web-based Learning Environments, Context, Context Metadata, Share and Reuse of Context, TEL Indicators

Résumé

L'intérêt pour l'observation, l'instrumentation et l'évaluation des systèmes éducatifs en ligne est devenu de plus en plus important ces dernières années au sein de la communauté des Environnements Informatique pour l'Apprentissage Humain (EIAH). La conception et le développement d'environnements d'apprentissage en ligne adaptatifs (AdWLE - Adaptive Web-based Learning Environments) représentent une préoccupation majeure aujourd'hui, et visent divers objectifs tels que l'aide au processus de réingénierie, la compréhension du comportement des utilisateurs, ou le soutient à la création de systèmes tutoriels intelligents. Ces systèmes gèrent leur processus d'adaptation sur la base d'informations détaillées reflétant le contexte dans lequel les étudiants évoluent pendant l'apprentissage : les ressources consultées, les clics de souris, les messages postés dans les logiciels de messagerie instantanée ou les forums de discussion, les réponses aux questionnaires, etc.

Les travaux présentés dans ce document sont destinés à surmonter certaines lacunes des systèmes actuels en fournissant un cadre dédié à la collecte, au partage et à la réutilisation du contexte représenté selon deux niveaux d'abstraction : le contexte brut (résultant des interactions directes entre utilisateurs et applications) et le contexte inféré (calculé à partir des données du contexte brut). Ce cadre de travail qui respecte la vie privée des usagers est fondé sur un standard ouvert dédié à la gestion des systèmes, réseaux et applications. Le contexte spécifique aux outils hétérogènes constituant les EIAHs est représenté par une structure unifiée et extensible, et stocké dans un référentiel central. Pour faciliter l'accès à ce référentiel, nous avons introduit une couche intermédiaire composée d'un ensemble d'outils. Certains d'entre eux permettent aux utilisateurs et applications de définir, collecter, partager et rechercher les données de contexte qui les intéressent, tandis que d'autres sont dédiés à la conception, au calcul et à la délivrance des données de contexte inférées.

Pour valider notre approche, une mise en œuvre du cadre de travail proposé intègre des données contextuelles issues de trois systèmes différents : deux plates-formes d'apprentissage Moodle (celle de l'Université Paul Sabatier de Toulouse, et une autre déployée dans le cadre du projet CONTINT financé par l'Agence Nationale de la Recherche) et une instanciation locale du moteur de recherche de la fondation Ariadne. A partir des contextes collectés, des indicateurs pertinents ont été calculés pour chacun de ces environnements. En outre, deux applications qui exploitent cet ensemble de données ont été développées : un

système de recommandation personnalisé d'objets pédagogiques ainsi qu'une application de visualisation fondée sur les technologies tactiles pour faciliter la navigation au sein de ces données de contexte.

Mots-clés : Environnements Informatiques pour l'Apprentissage Humain (EIAH), adaptation, contexte, données d'observation, partage et réutilisation

Contents

List of Figures

List of Tables

INTRODUCTION

Chapter 1 - General Introduction

Contents

1.1 Scientific Context

This manuscript presents the summary of four years of work conducted as part of a thesis. This work took place within the SIERA[1] research team of the IRIT[2] laboratory in Toulouse. IRIT is a joint research unit (UMR[3] 5505) common to several institutions: University of Toulouse III Paul Sabatier[4], CNRS[5], INPT[6], and the Social Sciences University of Toulouse I, Capitole[7]. A priority of the laboratory is the development of transversal projects open to national and international communities to promote and disseminate the technical and scientific culture, to transfer knowledge through the organization of regular research/socio-economic world meetings and events dedicated to the general public and schools, and finally, to transfer the know-how and technologies through the creation of joint laboratories with industry.

[1] Service IntEgration and netwoRk Administration
[2] Institut de Recherche en Informatique de Toulouse (Toulouse Computer Science Research Institute) - http://www.irit.fr
[3] Unite Mixte de Recherche (Mixed Unit of Research)
[4] http://www.univ-tlse3.fr
[5] Centre National de la Recherche Scientifique (Scientific Research National Center) - http://www.cnrs.fr
[6] Institut National Polytechnique de Toulouse (Toulouse National Polytechnical Institute) - http://www.inpt-toulouse.fr
[7] http://www.ut-capitole.fr

Research topics of the SIERA team, led by Prof. Abdelmalek Benzekri, focus on the control and management of last generation infrastructure and communication services, but also of dynamically aggregated, distributed and cross-organizational, complex applications and systems. Thus, the work attempts to define and evaluate new management paradigms, and provides the community with architectures, platforms, tools and contributions to standardization. In this context, several themes are subject of specific studies and proposals: (1) a design process of management applications offering a double independence, one regarding the specific areas to manage and the other regarding the management platforms, (2) evaluating the potential of SMA[1] to meet the requirements of cooperative and autonomous management (or self-management), (3) the security management of information flows of a virtual organization through the refinement of policies for building and deploying cross-domain policies, and the definition of a management environment able to control the security policies and formally validate established security rules linked to the appropriate mechanisms and security protocols, and (4) the distribution and management of online learning environments that represent nowadays complex systems for the exploitation of learning objects on a large scale. The subject of this thesis is integrated within the last theme. This document presents the work we performed and the results we obtained.

1.2 Research Context

Its fast development, the increased popularity and ease of use of its tools, made the World-Wide Web the most important media for collecting, sharing and distributing information [Zai01]. Many organizations and corporations use the web to provide various information and services such as customer support or on-line shopping. It is not surprising that the web is the architectural choice for any modern advanced distance educational systems.

The educational systems through which students acquire skills and knowledge, usually with the help of teachers or facilitators, learning support tools and technological resources, and use the web as a mean of content delivery and knowledge sharing, are called *Web-based Learning Environments* (WLE) [ASS+03]. Typical WLEs include course content delivery tools, synchronous and asynchronous communication systems, quiz modules, virtual workspaces for sharing resources, white boards, grade reporting systems, assignment/submission components, etc.

The benefits of traditional web-based education have been demonstrated [Bru98]: classroom independence and platform independence. An application installed in one place can serve thousands of students all over the world equipped with an Internet capable device. The problem of these systems is that they are nothing more than a network of static web pages where the same content is provided independently of the target learner [Bru98]. Learners from different backgrounds are given the same

[1] Service Monitoring Agent

learning content at the same time, even if they may only be interested in a small part of the whole learning content.

To overcome the user independence of traditional WLEs, advanced web-based learning applications able to adapt their content to every learner needs have appeared. With *Adaptive Web-based Learning Environments* (AdWLEs), it is possible to do better than to align learners to the same learning content; it is possible to actively engage the learner with a teaching strategy and material that appeal to his/her prior knowledge, needs, goals, motivations and learning style preferences [ST03]. As a consequence, students will be able to achieve learning goals more efficiently when pedagogical processes are adapted to their individual differences [Fed00].

1.3 Research Issue

Within traditional face-to-face learning sessions, teachers know their students in terms of knowledge, motivation, learning progress or learning style because they interact with them inside a classroom or during one-to-one tutoring sessions. They can further exploit this information to adapt or optimize their instructions by adding an explanatory sequence or a complementary exercise, or leaving an exercise for the next session. Within online and computer-based learning environments, the observation and evaluation of a learning session are often based on the analysis of data automatically collected during or after the learning session. Therefore, a prior step for adaptive systems to support users during their pedagogical process is to capture the context in which they operate, and thus to collect a large amount of data resulting from the interactions of users with the adaptive systems and educational resources. Then, AdWLEs reuse the collected contextual data to build user profiles and to further match these profiles with a pool of resources and activities using various adaptation algorithms to find out what should be provided next to a specific user.

However, considering the popularity of the social web (or Web 2.0) over the recent years, users operate in a much larger computational context while learning, using non-learning tools and services such as instant messaging, blogs, social networks, etc. By capturing information resulting from interactions between users and all these tools, context information and thus user profiles would be much more accurate, detailed and precise. Also, by providing various adaptive systems with the opportunity of accessing this pool of data, they would be able to set up enhanced adaptation techniques and algorithms based on data describing users in wider environments, instead of reusing their single source of observational data only.

The following scenario highlights the need to federate multiple sources of observation data. The teacher creates his course on Moodle. Afterwards, he deploys the course content together with the associated laboratory subject. He proposes to his students to perform the laboratory activity using a

remote-lab platform. The students consult the course content together with the laboratory subject on Moodle, and perform the lab using the remote-lab platform. Moodle has already a mechanism to observe the interactions of their users. Nevertheless, the remote-lab platform usually does not have this option as shown by [BVB11a]. If we are able to federate the two sources of observation data, we can give the possibility to the teacher to evaluate his course, and to evaluate and provide live support to his students, using only one application. The students can also beneficiate from collaborative learning.

Therefore, the following research issue arises: *how heterogeneous data issued from various sources can be represented in a unified way, and how this information can be easily shared and reused at a large scale?*

To address the above question, our approach stands on two main proposals: a context metadata model able to federate heterogeneous data, and supported by an open architecture. From the modeling point of view, several objectives must be achieved:

- On the one hand, the context model should not be restricted to specific context elements, but rather support various entities describing contexts of multiple applications and tools. On the other hand, semantic constraints should reduce the number of admissible combinations between entities but also the range of admissible values to reflect precise context information;
- The model should provide various abstraction levels to represent high level context information that is common to any system or application, but also low level data very specific to a given tool or resource;
- The model should allow the easy integration of new context, since new applications may have to be observed;
- The model should be simple to interpret and exploit: a rich but hard to use model is often unusable.

To effectively support the model, we propose to design an architecture characterised by the following capabilities:

- A central repository to store data represented according to the context model. This repository has to manage a large amount of data to ensure scalability, since it acts as the context storage component of a large number of adaptive systems;
- A set of tools and services to ensure the easy indexation of context metadata into the repository, retrieval of this data, and extension of the model to promote integration of a wide range of application into the framework;

- Sensors dedicated to the gathering of context metadata from adaptive systems, together with adaptive components responsible for adaptation process. These components should not be tightly coupled with the above tools and services to facilitate their integration into existing applications;
- As sensitive data related to personal information about users are stored into the repository, the designed architecture must comply with current legislation about privacy of users [Com95] and thus propose some organisational principles and/or guidelines to ensure the confidentiality of private data.

1.4 The Structure of the Manuscript

To present our proposals, this manuscript is divided into three main parts: (1) a presentation of nowadays AdWLEs followed by a state of the art covering the various components and initiatives or projects involved in the collection and storage of context data, to highlight the borders to be crossed in order to reach the goals described above, (2) the solutions we propose to facilitate the share and reuse of this data, and (3) the validation of our proposals through their deployment within different existing e-learning environments. Finally, some conclusions and future works end this document.

1.4.1 Part I: State of the Art

The first part comprises two chapters. The first one presents the transition from static WLEs to learner-oriented adaptive WLEs in order to highlight the additional components required for adaptation, and express the need for AdWLEs to collect the context of the user. The second chapter exposes an analysis of several existing frameworks dedicated to the collection and storage of context data, and highlights their limitations.

Adaptive Web-based Learning Environments

Chapter 2 first identifies the components that are specific to adaptive WLEs: **(1) the learner model** composed of the domain dependent (i.e. the knowledge level of the learner about the topics of the domain to be taught) and independent (i.e. demographic, previous background, learning style, interests, goals) characteristics of the learner, **(2) the content model** which maps learning resources to domain concepts, and **(3) the tutoring model** which incorporates the adaptive techniques while defining what can be adapted, as well as when and how it is to be adapted. Then this chapter focuses on three classes of adaptive web-based learning systems: mirroring/monitoring/guiding tools, intelligent tutoring systems, and personalized recommendation systems. All these systems require a large amount of data

resulting from the interactions between users and systems. The systems able to collect contextual information about the users are called Context-aware Systems; they are presented in the next chapter.

Context-aware Systems

Chapter 3 defines modeling, architectural and privacy design principles that Context-aware frameworks have to comply with in order to promote the share and reuse of encapsulated data:

- **Model design.** The context model should be designed at two levels: the raw level directly collected from observed applications, and the inferred level based on the raw level. The modeling approach must also be able to take into account data specific to heterogeneous applications;
- **Architectural design.** Context-aware systems should fulfill architectural requirements such as openness (to facilitate integrations of new collecting sensors and adaptive components), repartition/distribution (to promote share and reuse of heterogeneous context data), and scalability (to offer access to context data to widely distributed and disparate applications);
- **Privacy design.** The seven principles of privacy established by the European Commission related to the protection of personal data have to be addressed by the context-aware systems before rolling out within companies.

After defining these principles, Chapter 3 presents an overview of ten existing context-aware frameworks. A deep analysis of these approaches is further exposed to see how they meet the requirements we previously defined and to highlight the drawbacks that still remain to be tackled. Thus, in Part II, we present our proposals to fulfill the identified limitations.

1.4.2 Part II: Share and Reuse of Context Metadata Resulting from Interactions between Users and Heterogeneous Web-based Learning Environments

This part is also composed of two chapters. The first chapter presents our approach for modeling context whereas the second chapter is dedicated to the global architecture supporting our models.

Modeling the Web-based Learning Environments

Chapter 4 presents our modeling approach that takes into account the design principles established in the previous chapter. Our models are based on the Common Information Model (CIM) proposed by the DMTF dedicated to the management of applications, systems, services and networks. The raw level comprises (1) the user context describing a user from a learner, teacher or tutor point of view, (2) the environment context representing the learning systems and resources, and (3) the usage context to link

the two previous models to describe the activities performed by a user on a system or resource. The inferred level process on this raw data to derive more abstract measures through the form of significant indicators, and distinguishes elementary, arithmetic, and complex indicators.

The Global Architecture

Chapter 5 introduces our architecture for collecting, storing and sharing context data. This architecture is inspired from the DMTF architecture and comprises four layers: (1) the context layer embeds a repository to store data together with a set of management components, (2) the intermediate layer provides AdWLEs with a set of services to easily manipulate data of the context layer, (3) the AdWLE layer is composed of the systems and tools from which context information is collected and reused, and (4) the user layer where sensitive context data is stored to ensure users' privacy; moreover, some functional principles and guidelines are also presented.

1.4.3 Part III: Implementation

The third part presents the implementation of the models and architecture exposed in Part II and comprises three chapters. The first chapter gives an overview of the technologies and formats used to implement our architecture, the second one presents some AdWLEs integrated within our framework together with their context models, while the third one exposes some applications and tools that exploit the collected data.

Adopted Technologies and Formats

Chapter 6 depicts the technologies used to implement our architecture, together with the formats adopted by the intermediate layer to insert, query and expose context data. The tracking layer is implemented by an Object-Relational database which combines the advantages of both relational and object-oriented paradigms, whereas the middleware layer is implemented as a service-oriented architecture.

Furthermore, the AdWLE layer from which contextual data is collected integrates a Moodle server (deployed within the IUT A Paul Sabatier), a specific tool built for a CONTINT project funded by the French National Research Agency, but also a local instantiation of the ARIADNE Finder search tool. Finally, the user layer suggests two approaches to store the sensitive data: into a local DMTF implementation natively integrated within the Microsoft Operating System, and into cookies.

Sources of Context Metadata

Chapter 7 focuses on the context metadata (i.e. data about the context in which the user operates) collected during our experimentations. Step by step, we show how our models evolve to represent context specific to each application by adding new classes into the environment and usage context; at the end, the model is able to federate context produced by three heterogeneous applications. The unified view of users' context among various applications is meant to enhance the adaptation process, since it represents the interests of the users over a multitude of applications. Based on the collected context, we inferred various indicators, from simple ones such as the total number of activities performed over a learning resource, to complex ones such as the Proportion of Actions Indicator (PAI).

Applications Built upon our Framework

Chapter 8 presents two different tools that show how the context data can be reused. The first application is a personalized recommendation system able to recommend learning resources according to (1) context metadata gathered when a user performs an action on a resource, (2) the user and document models described in terms of semantic annotations based on the ACM taxonomy, and (3) an algorithm able to calculate similarities between these models. The recommendation service that implements this algorithm offers synchronous recommendations to online users.

The second application reusing the context data is MultiTravle (Multi-touch TRAcking VisuaLization systEm), a context visualisation application based on multi-touch technology. This application interacts with the services of the middleware layer to retrieve and display the enclosed context in a generic, intuitive and ergonomic way. The visualization is performed on three levels: the level of systems, the level of resources, and the level of users.

The manuscript ends with the conclusions we have drawn after these years of thesis, and suggests some future works considered as a continuation of this labor.

PART I - State of the Art

Chapter 2 - Web-based Learning Environments

Contents

2.1 Introduction

Internet technologies have had a significant impact on learning industry in the past few years. Traditional institutions of higher education, universities and colleges have well understood the potential impact of these technologies, and the majority of them adopted e-learning within their own institutions as a means of delivering web-based courses to distant learners, or as support tools for courses delivered in the traditional classroom.

A web-based learning environment (WLE) is a set of teaching and learning tools designed to enhance student's learning experience by including Internet-based applications into the learning process. The main components of a WLE include curriculum mapping (breaking curriculum into sections

that can be assigned and assessed), student tracking, online support for both teacher and student, electronic communication (e-mail, forums, chat, web publishing), and hyperlinks to external resources.

2.2 Components

The main components of a web-based learning environment are described in Figure 2.1. The basics of a WLE are the Learning Objects (LO) which are defined as any document (a web page, an image, a simulation, a test) used for learning (red spots on Figure 2.1). These LOs are (re)used by teachers to create courses via Learning Management Systems (LMSs). The following sections will detail these components.

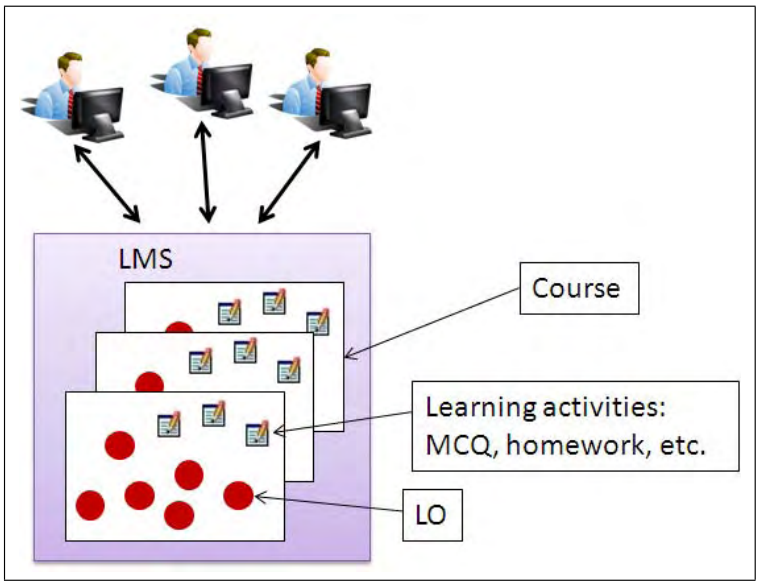


Figure 2.1: A traditional WLE.

2.2.1 Learning Objects

The origin of the term *Learning Object* is attributed to Wayne Hodgins, Director of Worldwide Learning Strategies Autodesk[1]. He had the intuition of a learning environment built on the basis of autonomous elements in 1992, as he watched children playing with Lego™ blocks. Following his presentation in different forums, several groups became interested and began to explore this concept both in U.S. and Europe. These LOs are currently the main focus of many works both in educational institutions and standardization organizations. The idea is to define components or pedagogical bricks to facilitate their aggregation, sharing and reuse.

[1] http://www.autodesk.com

2.2.1.1 Definition

The IEEE[1] LOM[2] P1484.12 working group gives two definitions for learning objects. The first definition is in the final version describing the Learning Object Metadata (LOM): *a learning object is defined as any entity, digital or not, which can be used for learning or teaching*. This definition includes everything as a learning object. The second definition is on the LOM standard presentation page: *a learning object is defined as any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning (computer-based training systems, interactive learning environments, intelligent computer-aided instruction systems, distance learning systems, and collaborative learning environments)*. This last definition restricts learning objects to entities used in computer-based learning systems.

In practice, a learning object can be a web page, an image, a simulation, a test or any other element involved in the learning process. Learning objects are not limited to a course or learning content, they may refer to procedures or guidelines to help learners during their learning process.

2.2.1.2 Learning Objects' Granularity

The Lego™ metaphor has highlighted the concept of granularity. The Autodesk company defines 5 different levels of granularity [Hod02]:

- **Raw data elements** are located at the lowest granularity level and refer to content of the data level. Examples include single sentence or paragraph, illustration, animation, etc.;
- **Information objects** are composed of raw data elements in order to create granular, reusable chunk of information. They explain a concept, illustrate a principle or describe a procedure;
- **Application objects** are a set of assembled information objects which focus on a single learning objective;
- **Aggregate (or assembly)** extends to larger learning objectives (lessons or chapters);
- **Collections** represent the highest granularity level, and depict courses, books, movies, etc.

Insofar, as learning objects are very diverse and pursue various objectives, there must be a way to accurately describe these learning resources: these are descriptors, or metadata.

[1] Institute of Electronical and Electrics Engineers - http://www.ieee.org
[2] Learning Object Metadata - http://ltsc.ieee.org/wg12

2.2.1.3 Metadata

According to IEEE-LTSC[1], metadata is data describing other data. It comprises a set of descriptors and can be structured regarding a given schema. Another definition is given by [Sim02]: *a metadata represents additional information to a text or software in order to provide details on its content*. In the case of a computer file, the name, size or creation date represent metadata. Metadata ensures several functions [NIS04]:

- Unique **identification** of a resource;
- Resources **discovery**. They can be searched according to different criteria, identified, localized or aggregated around common criteria;
- **Classification** of resources based on audience or subject;
- **Interoperability**, thanks to various descriptions which can be understood by both human and machine;
- **Archivation** and **preservation** warranty which ensures that the resource will not be forgotten and that it will still be accessible in the future.

In the e-learning area, metadata supports effective identification of educational resources. The main objective of metadata is to allow various software or e-learning platforms to interpret the "descriptive file" of a resource, and to be able to treat the resource in accordance to its description. Then, some standards emerged: the most popular ones are *Dublin Core (DC)* and *Learning Object Metadata* (LOM), the later being considered as the dominant standard in the field.

2.2.1.3.1 Dublin Core

The Dublin Core is a generic metadata schema for describing digital or physical resources, and appeared as a reaction to the problem of finding resources on the growing World Wide Web [Stu09].

The Dublin Core emerged after a working group meeting in 1995 in the city of Dublin, in the U.S. state of Ohio, to define a common core of elements used by the U.S. government to describe digital resources of official records (defense, justice, etc.). Currently, eighty elements constitute the DC standard. All elements are optional, and defined independently of each other. Dublin Core allows any number of combinations of elements to be used to describe a resource. The Dublin Core standard is maintained by an international multi-disciplinary working group, the DCMI (Dublin Core Metadata Initiative)[2] bringing together librarians, computer or museum specialists, researchers or practitioners from public or private organizations.

[1] IEEE Learning Technology Standards Committee - http://www.ieeeltsc.org/
[2] http://dublincore.org/

2.2.1.3.2 Learning Object Metadata

IEEE Learning Object Metadata standard has its origins in earlier work within the European Project ARIADNE and the IMS Global Learning Consortium, beginning in 1995. In June 2002, the IEEE Standard Organization finally approved LOM as an international standard used to describe a learning object and similar digital resources used to support learning. LOM consists of a single hierarchy of seventy six elements divided into nine categories (general, life cycle, meta-metadata, technical, educational, rights, relation, annotation and classification).

The LOM standard allows to extend and add new data elements as required by applications, thus creating so called "application profiles"; LOM-FR[1] is an application profile developed for education in France first published in 2006. This flexibility in the standard has encouraged metadata developers to use IEEE-LOM over Dublin Core as the base standard for developing new application profiles that suit their application needs [AKD06].

2.2.2 Knowledge Pools

Knowledge pools, or Learning Objects Repositories (LOR), are systems in which learning objects are indexed and stored according to a metadata format. They often use harvesting protocols to gather metadata, the most popular being OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting)[2] [LS01]. It allows to create and automatically update centralized warehouses where metadata from various sources can be queried simultaneously. Used especially by the Open Archives and institutional repositories, it is now widely used in cultural heritage institutions, including libraries.

The ARIADNE infrastructure is an example of LOR. It is a distributed network of repositories that encourages the sharing and reuse of learning objects [TVP+09], and comprises several components (see Figure 2.2):

- **Repositories** store both metadata and learning objects. The repositories are enhanced with three interfaces: a search interface based on the Simple Query Interface (SQI) [SMVA+05] specification, a publishing interface based on the Simple Publishing Interface (SPI)[TMVA+08], and a harvesting interface based on OAI-PMH;
- **The federated search engine** offers transparent search to a network of repositories (including the Global Learning Objects Brokered Exchange - GLOBE[3]) using SQI;
- The **harvester**, built on OAI-PMH, harvests metadata from target repositories (ARIADNE currently harvests from more that twenty repositories);

[1] http://www.lom-fr.fr/
[2] http://www.openarchives.org/pmh/
[3] http://www.globe-info.org/

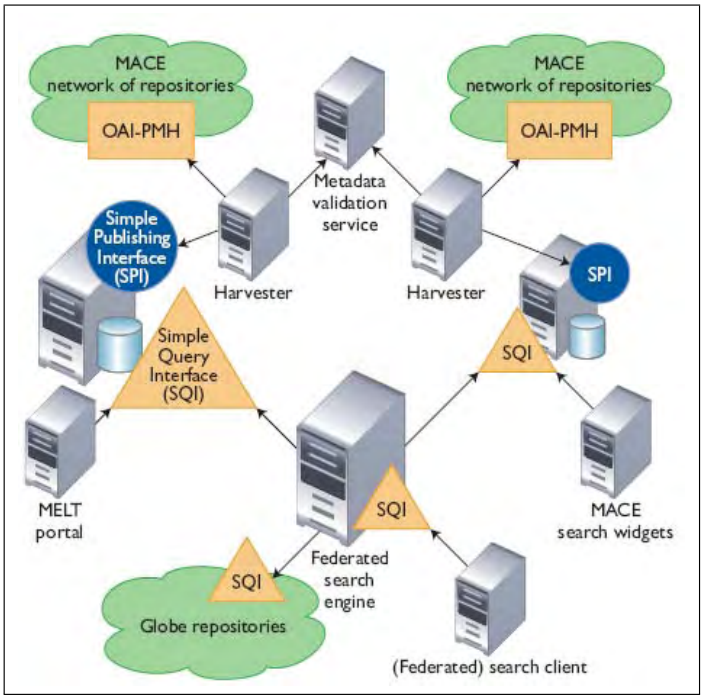


Figure 2.2: Connecting repositories through federated search and harvesting [TVP+09].

- The **metadata validation service** validates harvested metadata against a specific schema;

- **Search Clients** are graphical interfaces which allow users to search educational content and browse results. Examples of such clients are the federated search client (or Finder)[1], or the Metadata Ecology for Learning and Teaching (MELT) portal[2].

Infrastructures such as ARIADNE are able to federate an important number of learning objects spread all over the world. However, the objects cannot be provided to students in an ad-hoc manner, there is a need for organizing and structuring the pedagogical scenario.

2.2.3 Pedagogical Scenarios

A pedagogical scenario (or learning scenario, or learning design) defines the activities which must be completed by learners, the sequencing of these activities as well as the mapping of learning objects to these activities in order to achieve predefined outcomes [PV05]. Two standards for designing pedagogical scenarios emerged.

[1] http://ariadne.cs.kuleuven.be/finder/ariadne/
[2] http://info.melt-project.eu/

2.2.3.1 IMS-LD

Using the LOM standard only to describe learning resources is inadequate to describe the wide variety of learning situations encountered on the field [RvRK+02].

Based on this observation, a lot of works proposed formalisms to *a priori* describe learning situations instrumented by communication and information technology. Thus, the Educational Modeling Languages (EMLs) emerged. An EML is defined by the European Standardization Committee as *a model of information and semantical aggregation, describing the contents and processes involved in a learning unit according to a pedagogical perspective and to ensure the reusability and interoperability*. The abstract term "learning unit" refers to any granularity of learning resources (course, lesson, module, etc.).

In this context, the North American IMS consortium began studying a specification of such a language, and created in February 2003 the Learning Design specification V1.0 (IMS LD). This proposal, largely inspired from the EML developed by [KM04], provides a conceptual framework for modeling a learning unit and claims to offer a good compromise between genericity allowing to implement variate pedagogical approaches on the one hand, and expressive power offering a precise description of each learning unit on the other hand [Lej04].

2.2.3.2 SCORM

The Sharable Content Object Reference Model (SCORM) is a specification for content packaging introduced by the Advanced Distributed Learning Initiative (ADL)[1] [CHSW04]. SCORM promotes the portability of learning content from one Learning Management System (LMS) to another through packages, as well as the reusability of learning objects allowing instructional designers to distribute their content on a wide variety of e-learning platforms without rewriting efforts.

SCORM 1.1 was the first production version, but lacked a robust packaging method and support for metadata. It was quickly abandoned in favor of SCORM 1.2, the first version widely used and supported by most nowadays WLEs. SCORM 2004 is the current version.

The main difference between SCORM and IMS LD is that the SCORM initiative places learning objects as the centric elements of the learning process, while in the case of IMS LD the centric elements are the activities associated to the learning objects [Per04].

The pedagogical scenarios are often provided to learners through learning management systems; scenarios are also called courses in these systems.

[1] http://www.adlnet.org/

2.2.4 Learning Management Systems

Learning Management Systems (LMSs) are crucial within a WLE, because they represent systems by which learners study courses created by teachers or trainers. [Pau02] defines an LMS as a *software that automates the administration of training events. The LMS registers users, tracks courses in a catalog, and records data from learners; it also provides reports to management. An LMS is typically designed to handle courses by multiple publishers and providers. It usually doesn't include its own authoring capabilities; instead, it focuses on managing courses created by a variety of other sources*. A multitude of platforms have been developed within public or private institutions in order to answer specific needs of learning, thus making various services and functionalities available to e-learning actors.

Even if standards such as SCORM and IMS LD promote the portability of learning content from one system to another, LMS remain closed systems that are not, or poorly, opened to other components of a WLE. Therefore, in order to allow exchanges of learning content between LMS and LOR, some works [BV05] suggested an intermediate layer making possible the communication between these two kinds of systems.

However, the content delivered to learners remains the same for all students enrolled into the class [LB11]. But students are different in age, culture, education background, way of learning, knowledge, attention, interests, etc. It is thus of vital importance to provide them with learning contents and teaching strategies that suit their individual needs.

2.3 Adaptive Web-based Learning Environments

Learners from different backgrounds are given the same learning content at the same time, even if they may only be interested in a small part of the whole learning content. Therefore, one challenge is to develop advanced web-based learning applications which adapt their content to every learners' need. With adaptive web-based systems, it is possible to do better than to align learners to the same learning content: it is possible to actively engage the learner with a teaching strategy and material that appeals to his/her prior knowledge, needs, goals, motivation and learning style preferences [SZR12]. As a consequence, students will be able to achieve learning goals more efficiently, since pedagogical scenarios are adapted to their individual differences [SEBAM10].

A very comprehensive definition of an adaptive learning environment is provided by [PLR04]: *a learning environment is considered adaptive if it is capable of: monitoring the activities of its users; interpreting these on the basis of domain-specific models; inferring user requirements and preferences out of the interpreted activities, appropriately representing these in associated models; and, finally, acting upon the available knowledge on its users and the subject matter at hand, to dynamically facilitate the learning process.*

2.3.1 Categories of Adaptation in Learning Environments

[PLR04] provides a high-level categorization instead of exhaustively enumerating all types of adaptations. This classification comprises four categories:

- **Adaptive interaction** refers to adaptations which take place at the systems' interface, and that are meant to facilitate and support the users' interactions with the system without modifying the learning content itself;
- **Adaptive course delivery** represents the most common collection of adaptive techniques used in learning environments. This type of adaptation is meant to taylor a course or a set of courses to an individual learner. The most typical adaptations in this category comprise dynamic course (re)-structuring, adaptive navigation support and adaptive selection of alternative course material;
- **Content discovery and assembly** refers to the application of adaptive techniques to discover and assemble learning materials coming from potentially distributed sources / repositories. An example of this category is the discovery of learning content from Internet;
- **Adaptive collaboration support** is intended to adapt the learning process that involve multiple actors, and consists in facilitating the collaboration tasks.

Compared to the traditional web-based learning environments, where the adaptation features are not taken into account, the adaptive learning environments enclose a number of components dedicated to this purpose.

2.3.2 Traditional WLEs vs Adaptive WLEs

Figure 2.3 presents a parallel between the components of traditional web-based learning environments and those specific to adaptive systems. In a traditional WLE, teachers or learning designers use learning management systems to build pedagogical scenarios on the basis of learning objects (sometimes retrieved through one or several learning object repositories). Thus, a traditional WLE may use two components: a LOR which contains the learning material to build pedagogical scenarios, and a LMS to deliver static scenarios to learners.

Compared to traditional WLEs, adaptive WLEs are continuously monitoring users activities in order to infer the needs of each student at any moment and apply some of the previous adaptation techniques. In order to ensure these tasks, beside the two components of traditional WLEs, an adaptive WLE requires some specific modules depicted on Figure 2.3:

- The **learner model** incorporates the characteristics of the learner. Two types of information is represented here: (1) domain independent data (demographic, previous background, learning

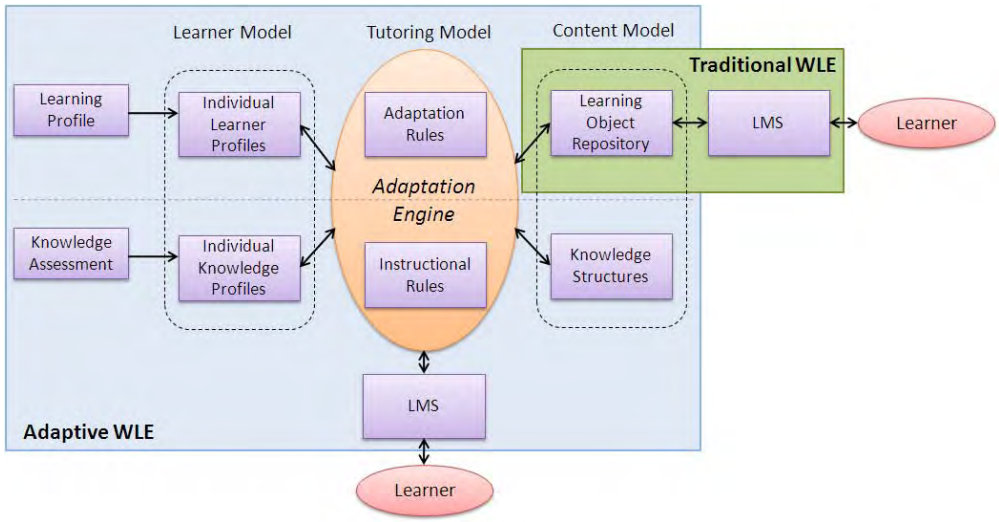


Figure 2.3: Traditional vs adaptive web-based learning environments (inspired by [ST03]).

style, interests, goals), and (2) domain dependent information which represents the knowledge level of the learner regarding the topics to be studied;

- The **content model** contains a learning object repository as a source of learning objects, and a domain model, or knowledge structure, which depicts the concepts related to the domain to be studied. Learning objects are then mapped to concepts in order to facilitate the process of pedagogical scenarios (re)engineering;
- The **tutoring model** represents the adaptive engine, and thus integrates the adaptive techniques. Specifically, it defines what can be adapted, as well as when and how adaptation must be achieved.

2.3.3 Components of an Adaptive Web-based Learning Environment

2.3.3.1 Learner Model

[DBAC04a] makes a distinction between user profile and user model, depending on how it is built within the web-based learning environments. Some systems ask users to fill a form about themselves (administrative, preference, etc.) when they register; the *user profile* is only updated when the user modifies the contained information. Other systems ask users to register, but for authentication purpose only. Further, the system monitors the users' behavior in order to create a *user model* representing the users' characteristics. Adaptation is performed according to the user model which is updated on the fly during the learning session.

Despite this, literature shows that user profile or user model are used for the same meaning: to specify the individual characteristics of the user. We adopt this convention on this manuscript as well.

It is expected that the recognition of students' learning needs, learning styles, preferences, as well as interests in specific learning modules will improve effectiveness of the learning process [MPG03]. Data describing a user model is illustrated on Figure 2.4 [MPG03]:

- **Knowledge** describes the level of understandings of the user about a concept [DBAC04a]. It is the single domain dependent component of the user model, and is seen as an overlay of the domain model [DBAC04b]. An AdWLE should be able to automatically update this information;
- **Goals** express learners' purposes. It is the answer to the question: what do learners want to achieve during the learning process? There are two kind of goals: long-term goals may refer to graduating a cursus, while short-term goals (called problem-solving goals [ND08]) may refer to learners' intention to solve a problem or exercise, acquire a skill or pass an examination;
- **Learning style** includes learning methods, particular to an individual, that are presumed to allow that individual to learn best. [FH95] claims that students' learning style can be defined by answering five questions. ***1.*** *What type of information does the student preferentially perceive: sensory sights, sounds, physical sensations, or intuitive memories, ideas, insights?* ***2.*** *Through which modality is sensory information most effectively perceived: visual pictures, diagrams, graphs, demonstrations, or verbal written and spoken words and formulas?* ***3.*** *How does the student prefer to process information: actively through engagement in physical activity or discussion, or reflectively through introspection?* ***4.*** *How does the student progress toward understanding: sequentially in a logical progression of small incremental steps, or globally in large jumps, holistically?* ***5.*** *With which organization of information is the student most comfortable: inductive facts and observations are given, underlying principles are inferred, or deductive principles are given, consequences and applications are deduced?* [FC00] considers learning styles as particularly important variables that influence the success of learning;
- **Interests** describe the short and long-term interests of a learner. Examples of long-term interests may be history, literature, while short-term interests may include interwar period, adventure novel, etc. Usually, the long term interests are provided by the user during the registration process, while the short-term interests are deduced by the system during users' interactions with the system;
- **Background** includes skills and knowledge that the learner gained in the past, and represents an important factor that influences the adaptive process;
- **Demographic** includes administrative information about the user such as name, surname, birthday, gender, postal address, e-mail, telephone number, etc.

The six student model dimensions thus compose an hyperspace experience of the user which is further exploited by the adaptation engine to provide learners with adequate content. However, to

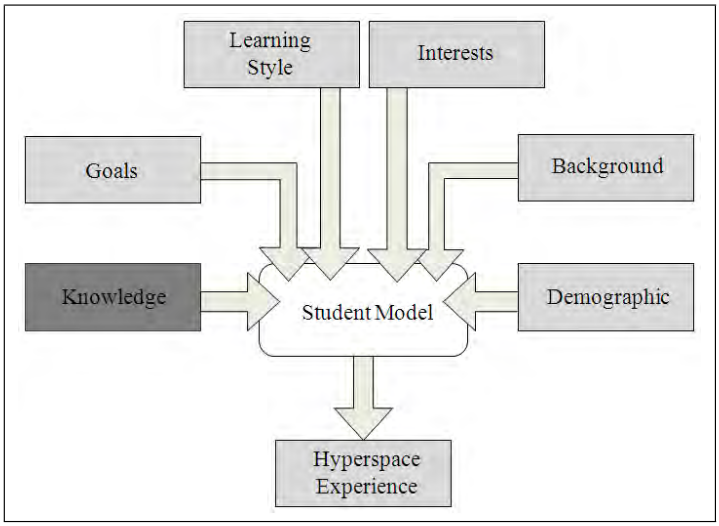


Figure 2.4: Generic student model, inspired by [MPG03].

generate and maintain user model, systems need relevant information about users. Therefore, user profile generation and maintenance require five design decisions [MLDLR03]: profile representation, initial profile generation, profile learning, source of relevant feedback, and profile adaptation.

The *profile representation* is the first point to focus on when designing adaptive web-based learning environments, since the other design decisions rely on it. According to [MLDLR03], profile representation techniques include, among others, history of selections, vector space model where the user profile is represented as a vector of features (usually words or concepts of interests) with associated values, weighted n-grams where the user profile is represented as a network of words and weights, classifiers such as neural networks, decision trees or Bayesian networks, and user-item ratings matrix.

Once a profile representation is defined, the adaptive system has to learn as much as possible about the user, and as soon as possible, to build the *initial profile*. The acquisition of the user profile can range from manual input or semi-automatic procedures to the automatic recognition by the system itself.

In case of automatic recognition, *profile learning* mechanisms are required. They include: vectors of features extracted from the documents the user interacted with (each document being represented as a vector of weighted features), clusters of users in order to match these clusters against actual information to conclude [XP01], or classifiers which receive as entry a data set and provide as output the interests of the user [AT01].

As the users' interests change over time, adaptive systems need up-to-date information (also called *relevance feedback*). This feedback can be either explicit (e.g. users are required to explicitly evaluate items to indicate how relevant is the item for them) or implicit (e.g. the system automatically infers users' preferences by monitoring their behavior).

Finally, *profile adaptation techniques* such as manual or gradual forgetting functions are required

to adapt the user profile to the new interests of the learner.

2.3.3.2 Content Model

The content model is composed of two entities: the learning object repository and the knowledge structure. While the former was presented in section 2.2.2, we focus here on the latter.

The knowledge structure (also called *domain ontology*) is constituted by a set of knowledge elements representing a concept, subject or topic of the domain. These concepts are organized as a hierarchical structure from high-level concepts (e.g. Analysis on Figure 2.5) to low-level concepts (e.g. L = Limits, D = Derivatives, I = Integrals, S = Series on Figure 2.5). Further, to each concept is associated a set of learning objects which are representative for that concept (e.g. LO1 for Limits, LO2 and LO3 for Derivatives, etc. on Figure 2.5). Finally, typical relations among concepts indicate a hierarchical relationship or a constraint between two concepts (e.g. B = IsPart_of, R = Requires, SO = SuggestedOrder, E = Explained_by on Figure 2.5).

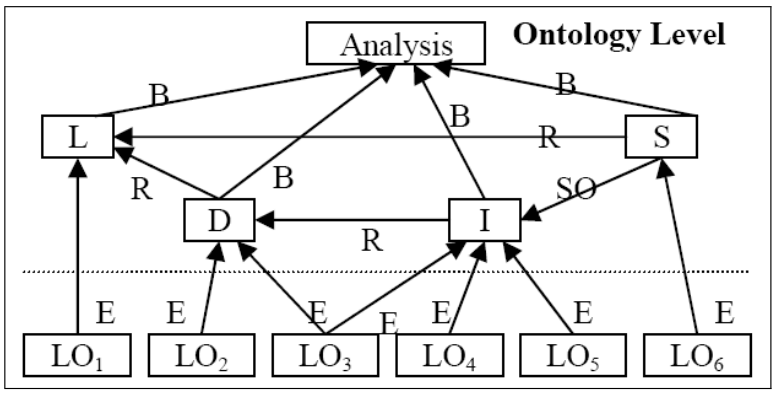


Figure 2.5: Example of knowledge structure [CGME02].

The building of the content model may be manual or semi-automatic, or even fully automatic (ensured by the system itself). While the first method was appropriate for enclosed systems (where the collection of documents is already known from the very beginning), it is not adequate in the context of nowadays distributed content delivery systems (where resource providers communicate together to exchange data) anymore.

Thus, the content model provides the basis for the following features [ST03]: assessments (what is the current status of a particular concept or LO?), cognitive diagnosis (what is the source of the problem?), and instruction (which LOs need to be selected next in order to fix a problem or introduce new topic?).

2.3.3.3 Tutoring Model

The way in which the system automatically adapts the presentation of information using prerequisites is known as *Tutoring Model* [DBHW99]. The tutoring model is composed of two sub-models: the *adaptation model* is common to all adaptive systems and provides users with adapted items or activities, whereas the *instructional model* is specific to an AdWLE and aims at guiding learners through the learning process.

The adaptation model, also known as *macroadaptation*, uses traditional Information Retrieval recommendation mechanisms such as content-based, collaborative-based or hybrid methods to exploit the domain independent components of the user model [TCLW06]. The idea behind the content-based method is that if a user liked an object in the past, he/she would probably like other similar objects in the future. Typically, a collaborative-based method matches people with similar interests and then recommends learning objects on this basis. On the other hand, an hybrid method attempting to combine the two previous techniques usually builds user profiles on the basis of content analysis, and then identifies users with similar preferences to perform collaborative recommendation.

The instructional model represents the logic (e.g. rules) that makes it possible for a student to navigate from a concept of the knowledge structure to another. On Figure 2.5, if the student gained a good score over the Derivatives concept, he/she can learn further the Integrals concept, otherwise the instructional model will provide him/her with additional resources related to the Derivatives concept. The instructional model, also known as *microadaptation*, usually contains a set of rules which exploits the domain-dependent component of the user model (knowledge) to identify which knowledge concept or skill element should be selected next from a pool of non mastered objects.

2.3.4 Some Classes of Adaptive Web-based Learning Systems

The variety of learning systems which adapts their behavior to the user is very large. In this section we present three types of such systems: AdWLEs based on inferred data, intelligent tutoring systems, and personalized recommendation systems.

2.3.4.1 AdWLEs Based on Inferred Data

Within online and computer-based learning environments, the observation and evaluation of a learning session are often based on the analysis of a large amount of data automatically collected during or after the learning session, and translating user-system interactions. The efficiency of the analysis depends on the quality of inferred data, or so-called *indicators*: they should match the teacher's observation needs in order to help him understand students' behaviors better. According to [Dim04], an indicator is a variable in a mathematical sense, which is assigned a value represented as a numerical,

alphanumeric or even graphic form, and characterized by a status; it can be raw (unit is undefined), calibrated (unit is defined) or interpreted (unit is implicit).

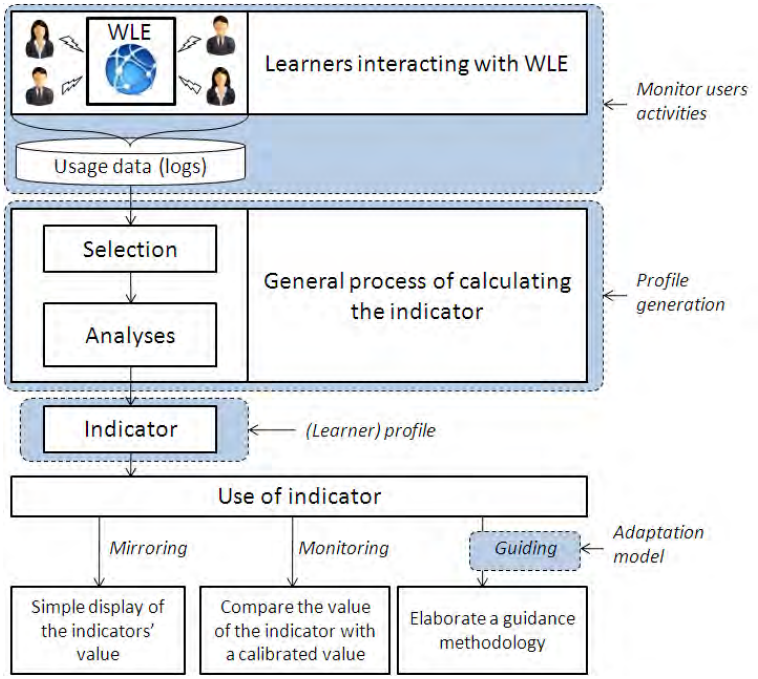


Figure 2.6: Indicators design, calculation and use [Djo11].

Indicators are formally described as the result of several transformations and abstractions of raw data collected during a learning session [NIC10]. The whole process of indicator design, calculation and use is depicted in Figure 2.6. Users interact with the WLE, and their activities are stored into a database. The selection of data among all existing usage data allows to (semi) automatically retrieve usage data of specific users. This information represents more or less complex indicators: a basic indicator refers to data that can be processed by a basic mathematical function (such as the total number of access to an online resource, the percentage of success while performing a quiz, etc.), whereas the calculation of advanced indicators results from complex algorithms and mathematical functions operated on basic indicators (the number of learners who interacted at least once with a given module during a given time interval, or the interaction rate of a given learner with respect to all interactions of all learners during a time interval, are examples of complex indicators [AKMF04]). Indicators are reused to provide simple or more complex feedback through mirroring, monitoring and guiding systems [SMJM05].

2.3.4.1.1 Mirroring Tools

The mirroring tools reflect the calculated indicators back to the user, for example as graphical visualization of actions or contributions. These systems are designed to raise students' awareness about

their behaviors. Learners and teachers are responsible for comparing the state of their learning outcomes with their own model of desired objective to determine what remedial actions are needed: the adaptation is not performed by the system, users are in charge of this process. Most often, indicators provided by mirroring tools are used to facilitate the task of learning scenarios re-engineering or to assist students when issues are encountered.

In ClassroomVis [FHMC07], viewing indicators in real time allows tutor to observe the synchronous activity of a group of learners and adapt this activity by interacting directly with the interface. Bubbles are used to represent activities, and links to represent the path between activities. Learners are represented by faces. These visualizations dynamically change over time, depending on the number of usage data produced by the learners in a time interval. If a learner is "sleeping" in a particular activity, the tutor can intervene by suggesting him to participate more actively.

Beside ClassroomVis, other learning systems are classified as mirroring tools. GISMO [MM04] is tracking learners activities in Moodle[1] and generates graphical representations that can be explored by course designers. In the same way, the Assistment system [FH05] helps teachers to assist students' development and assess their abilities within middle school mathematics classes through detailed reports about students and class level.

2.3.4.1.2 Monitoring Tools

Monitoring or metacognitive tools display information about what the desired information might look like alongside a visualization of the current state of indicators. A difference is made between a productive and unproductive value of an indicator [SMJM05]. A productive value corresponds to a representation of learners' interactions with the system that might positively influence learning: if we want learners to interact frequently we have to maintain a high value on a reciprocity indicator; if we want to participate equally we have to minimize the value on an asymmetry indicator. An unproductive value corresponds to interactions which are meant to interfere with productive ones: a change of subject in a chat, or unequal participation of students. Like mirroring tools, users are responsible for making decisions regarding diagnosis and improvement (i.e. adaptation).

Hop3x [NIC10] is a system helping third year graduate students to learn and practice Java programming. Starting from some predefined logged data, Hop3x calculates indicators such as the average time each student spent on each question, the program execution frequency each student performed, or the assimilation of past errors by a student. If the indicator "the average time each student spends on each question" returns a value upper than a reference value, teachers can modify the learning scenario for the next session or propose immediate solutions to help students better solve problems.

Jermann [JD08] has developed a system that displays participation rates of collaborators as they

[1] http://www.moodle.org

are solving a traffic light tuning problem. Indicators on the screen represent the number of messages posted by each student in the chat with respect to the number of problem-solving actions a student and his/her teammates performed. The system displays a color-coded model of desired interactions next to the actual value of the indicator. The students are using this color code to judge the quality of their interaction and appreciate whether or not to perform remedial actions.

2.3.4.1.3 Guiding Tools

The guiding systems compare the actual value of an indicator with an internal productive one, and offer automated advice intended to increase the effectiveness of the learning process. The desired value of the indicator and the current value are typically hidden from the student. Compared to mirroring and metacognitive tools (where adaptation is externalized and under the responsibility of learners and teachers), guiding tools perform adaptation usually through adaptation rules (see Figure 2.6).

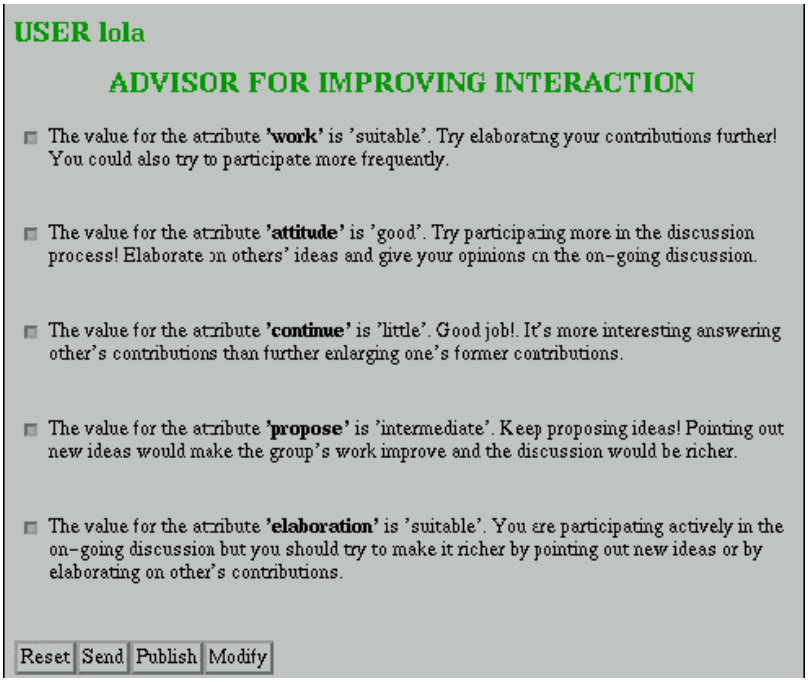


Figure 2.7: Examples of advising messages generated for a student by DEGREE [BF00].

MATHEMA [PGG09] is an interactive problem solving support system for senior high school students. Another guiding tool is DEGREE [BF00], a Computer-Supported Collaborative Learning (CSCL) system that can advise social aspects of interactions that occur within a group. To do this, DEGREE asks the users to select the type of contribution (i.e. proposal, question or comment) each time they contribute to the discussion. Their contributions are tracked and on their basis, two levels of social indicators are inferred. Low level indicators, directly calculated from the contributions and usage data, include *number-of-messages*, *initiative*, and *argumentation*, while high level indicators, calculated on

the basis of low-level ones, include *cooperation* and *work quality*. The values of the indicators range from *awful* to *very good*. On the basis of indicators values, the advisor offers tips to students to improve their collaboration (Figure 2.7 illustrates some examples of advises).

Beside AdWLEs which perform adaptation through computed indicators, a more complex class of adaptive learning systems is the Intelligent Tutoring Systems (ITS).

2.3.4.2 Intelligent Tutoring Systems

Let us imagine that each learner in a classroom has a personal training assistant who pays attention to the learners' needs, assesses and diagnoses problems, and provides assistance as needed. The assistant can perform many of the routine instructional interventions but alert the instructor of problems he/she cannot manage as well. Providing a personal training assistant for each learner is out of the budget of most organizations [OR00], but the introduction of a virtual training assistant is a good option.

An intelligent tutoring system is based on Artificial Intelligence techniques and is defined as any system which is capable of emulating an instructor behavior to support students as they acquire knowledge [CGCC06]. The instructor is not present, and the system assesses and guides learners as they learn different concepts. ITSs seek to reflect a method of teaching and learning based on a one-to-one interaction between student and teacher by means of [Neg98]: control of the learning level, control of course navigation, adaptation to available information, adaptation of the teaching methodology, explanation of errors, answer to the student questions, advice, etc. ITSs have been shown to be highly effective in increasing students' performance compared with traditional WLE [RBD+08].

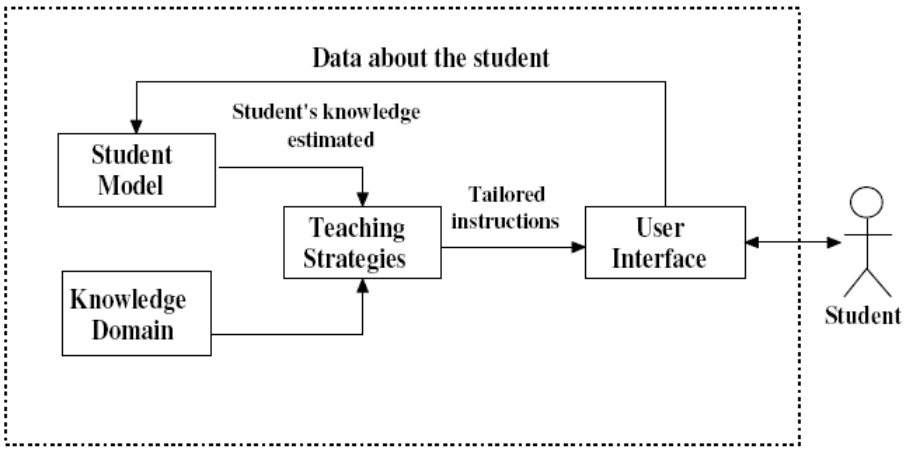


Figure 2.8: The main components of Intelligent Tutoring Systems [BHM06].

An intelligent tutoring system is composed of the following modules illustrated on Figure 2.8 [BHM06] [SK02] [JJG07]:

- The **student model** comprises the knowledge component of the learner model specific to AdWLE only, since ITSs are designed to handle a domain specific problem;
- The **knowledge domain** represents the content model of AdWLEs (see Figure 2.3), containing both a content provider and a knowledge structure, to map the LOs to domain topics;
- **Teaching strategies** define the Instructional Model specific to AdWLEs and refer to instructional methods for teaching. This component decides when to present a new topic, how to provide recommendation and guidance, and which topics to expose next, according to inputs from the student and knowledge models;
- The **graphic interface** implements the way the user interacts with the intelligent tutoring system. A well designed interface can maximise the capabilities of the ITS by presenting instructions to the student in a clear and direct way.

Two major types of ITSs are identified, depending on their objectives [BHM06]: curriculum sequencing and problem solving support.

2.3.4.2.1 Curriculum Sequencing

The purpose of these systems is to provide students with a personalized optimal path through the learning materials. This kind of ITS organizes domain concepts into a hierarchy of courses, modules, lessons, presentations, etc., which are related by prerequisites and other relationships [Mur99]. From the student point of view, the system looks like a traditional WLE, the difference residing in the dynamic sequencing of the content based on students' performance, the lesson objectives, and the relationships between course modules. The domain model is represented at a "superficial" level allowing any domain to be tutored, but the depth of the diagnosis and correction is limited by the abstraction of the domain model concepts. Curriculum sequencing systems are more appropriate for teaching conceptual, declarative and episodic types of knowledge, and less appropriate for pedagogically powerful, procedural or problem solving skills [Mur99].

Such a system is LIA (Learning Intelligent Advisor) [CGME02], which is in charge of the intelligent functionalities of an automatic Course Management System developed in the European Project "m-learning"[1]. [MM10] is another example to learn C++.

2.3.4.2.2 Problem Solving Support

The purpose of problem solving ITSs is to provide students with intelligent help at each step of a complex task (such as a project or a problem). When the student is stuck on one step, the system provides

[1] http://www.m-learning.org/

a hint showing the next correct solution, or offering appropriate feedback. These systems have a relatively deep model of expertise, thus the student is well supported to perform the next step or to complete the solution to the entire problem. As a consequence, the domain model has problem-solving specificity and is not (or hardly) transferable to another domain.

Oscar [LCM+10] is a conversational tutoring system which intelligently lead an online tutorial, miming the human tutor in offering students individualized problem solving support and solution analyses. Another problem solving ITS is Logiocando [LR07], which offers help to children of fourth level primary school to learn basic concepts of logic.

2.3.4.3 Personalized Recommendation Systems

The Personalized Recommendation Systems (PRS) usually take into account only the domain independent components of the user model in order to deliver personalized content (see Figure 2.9). These open systems usually exploit resources provided by open repositories (e.g. Internet, federation of repositories, etc.), and the adaptive content is dynamically built. They may use knowledge structures as a backbone to identify the topics related to a given course and perform automatic mapping between open resources and these topics. They use the conventional schema in which nodes and links are relatively unstructured (learners can freely access any node), and thus bring a larger degree of freedom in self-exploration but also generate significant disorientation.

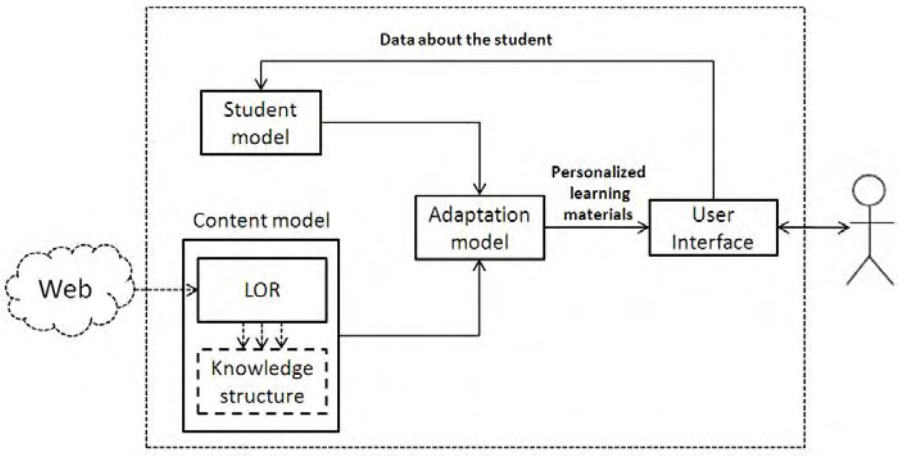


Figure 2.9: The architecture of personalized recommendation systems, inspired by [BHM06].

In a traditional adaptive e-learning system, learning material is personalized and delivered according to the learner model, and contents inside the system are *a priori* determined by the system designer. [TM05] suggests an evolving web-based learning system able to adapt itself not only to its users, but also to the open web: learning materials are automatically found on the web and further personalized and adapted based on the systems' observation of its learners. Other examples of personalized

recommendation systems are presented in [TCLW06], [Zai02] or [CDL04].

2.4 Conclusions

This chapter has drawn a parallel between the static traditional web-based learning environments which provide the same learning content to all students enrolled in a course, and the user-centric, adaptive web-based learning environments which take into account the differences between learners in order to adapt their learning content accordingly.

An important number of such systems has been developed in the recent years, and purposes of AdWLEs are various. Among these, we presented (1) the adaptive systems which calculate indicators and use these to improve the learning scenario, assist the learners when they encounter problems, or encourage students to participate more in the learning process through mirroring, metagognitive and guiding tools; (2) intelligent tutoring systems which provide a one-to-one tutoring to learners by emulating the tutor; and (3) personalized recommendation systems which recommend learning materials to learners based on their interests.

To perform adaptation, an AdWLE integrates a number of specific components: a learner model describes the characteristics of the learner, a content model specifies the topics of the expertise domain together with the resources to be taught, and an adaptive engine matches the learner model with the content model in order to provide adaptation.

The learner model is a key component of any AdWLE, since it represents the current experience of the user but also because the adaptive engine makes its decisions on its basis. In order to build the student model, the system needs to monitor the activities of its users, and thus to collect data resulting from their interactions with the system. Based on the collected data, AdWLEs build the student model by applying data mining techniques such as structured information retrieval, clustering or classification. A good learner model should be built on the basis of a large amount and detailed observational data expressing all interactions of the user with the system. In the literature, the observational data is also called *contextual data*. The systems specialized in collecting, storing and reusing contextual data are presented in the next chapter.

Chapter 3 - Context-aware Systems

Contents

After defining the *context* from our point of view, this chapter introduces important criteria that systems have to take into account to manage context, from both the modeling and architectural points of view. We then review the main existing approaches to see how they fulfill this set of criteria and to highlight some issues we will try to tackle in the next part of this document.

3.1 Context

Context is a complex concept that has been studied across different research disciplines, including computer science, cognitive science, linguistics, philosophy, psychology, and organizational science [AT11]. Some questions raised by [BB05] illustrate the complexity of context: *Is context a frame for a given object? Is it the set of elements that have any influence on the object? Is it possible to define context a priori or just state the effects a posteriori? Is it something static or dynamic? Some approaches emerge now in Artificial Intelligence [...] In Psychology, we generally study a person doing a task in a given situation. Which context is relevant for our study? The context of the person? The context of the task? The context of the interaction? The context of the situation? When does a context begin and where does it stop? What are the real relationships between context and cognition?*

3.1.1 Definition

In order to give answers to the above questions, [AT11] makes a synthesis of context definitions within disciplines related to computer science:

- **Data mining.** In this area, context is defined as those events which characterise the life phases of a customer and that can identify a change of his preferences or status (e.g. new job, marriage, divorce, retirement, etc.) [BL97];
- **E-commerce personalization.** The context is seen as the reasons of a customer to make a purchase (e.g. a book for improving his personal skills, a book as a gift, a book as a hobby) [PTG08];
- **Ubiquitous and mobile context-aware systems.** Within context-aware systems, context was initially defined as the location of the user (e.g. tour guides in the city [AAH+97]), the identity of people near the user, the objects around, and the changes in these elements. By time, other elements were added such as devices' light, motion, acceleration, touch, temperature, time or season;
- **Databases.** Context has been added to some DBMS[1] by incorporating user preferences and returning different results depending on the context at the time of queries' submission. In [SPV07], a set of contextual parameters is defined (e.g. location, weather), and users express their preferences on specific database instances (e.g. a restaurant) based on different values of the contextual parameters;
- **Information retrieval.** Most existing systems base their retrieval decisions on queries and document collections only, whereas information about search context is often ignored. The context is taken into account by means of query expansion or results filtering technics.

[1] Database Management System

As we can see, context is a multifaceted concept, but some authors tried to give a general definition [Dey01]: *any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.* Even if this definition is comprehensive, it introduces a series of incertitude such as "what does *any information* refer to?" or "what does a *situation* represent?".

3.1.2 A Formal Structure of Context

Based on Deys' definition, [ZLO07] divides this *information* into five categories illustrated on Figure 3.10: individuality, time, location, activity and relations.

The individuality context comprises anything that can be observed about an entity and is further clustered into four types refined categories:

- **Natural** entity context represents the characteristics of all existing things that occur naturally without human intervention (such as plants, stones, etc.). Interactions of human with these elements are comprised in this category as well;
- **Human** entity context covers the characteristics of human beings in terms of preferences: language, color schemas, menu options, etc.;
- **Artificial** entity context covers descriptions for any human-built thing like computers, vehicles, books, etc. Physical or chemical properties (temperature, pressure, sound, light, etc.) but also applications and services are included as well;
- **Group** entity context represents the common characteristics shared by a group of entities which interact with one another. Examples of groups are a cluster of computers, a library, etc.

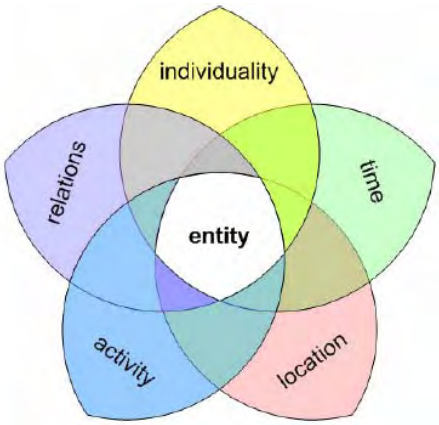


Figure 3.10: Five fundamental categories of context information [ZLO07].

Time is an important factor in understanding and exploiting context, since most statements are related over the temporal dimension. This category comprises elements such as the time when a customer made a purchase or downloaded a document, the time when a computer is turned on, etc. Beside the timestamp of an event, the periods of time (e.g. how many hours a PC was working, how many minutes a customer visited a web page of products) constitute a fundamental requirement of the context model. Thus, by analyzing the interaction history, usage habits of users can be inferred to predict future contexts.

Location became an important parameter with the recent development of mobile computing devices. This category describes physical or virtual residence of an entity. Moreover, the location can be either absolute, meaning the exact location of an entity, or relative, meaning the location of an entity relative to another entity.

The activity context covers activities an entity is (or will be) involved in, the final goal of that entity, but also the meaning of achieving it [ZLO07]. [RDL09] divides the activity context in 3 levels of hierarchy, from the lower to the higher: event, event block and task. Multiple events (e.g. letter typing) compose an event block (e.g. writing a paragraph), and multiple blokcs compose a task (e.g. writing a book).

Finally, the relation context express a semantic dependency between two entities and includes social relations (e.g. friends, enemies, neighbors, relatives, etc.), functional relations (e.g. using a hammer, operating a desktop computer, etc.) and compositional relations (e.g. the PC is composed of processor, RAM memory, etc.).

3.1.3 Our Vision about Context

Beside the different types of context, [ZLO07] suggests also a framework to establish semantical relations with other entities. Using this approach, Figure 3.11 represents our vision about context, specific to AdWLEs. We distinguish three different entities:

- **The user** (context) comprises the various characteristics of the user. The individuality attribute depicts the learner model as described in Chapter 2 (interest, knowledge, goals, etc.), while the other attributes refers to the current geographical location of the user, the time when the user logged into the system (for example), the learning activity the user is currently involved in, and the relations with the other two entities. This information is either directly filled with values, or inferred from the other two dimensions;

- **The environment** (context) comprises information about the system the learner interacts with, from the computational point of view. Let us mention that this entity does not represent the

content model of AdWLE, even if it relies on learning material. Indeed, this entity captures information about the learning resources which have been in the focus of the user, thus representing the attention of the users at any moment;

- **The usage** (context) describes what and how the user handled the environment. Beside the type of actions performed by users (e.g. consultation, download, etc.), the usage context contains also the devices used to perform these actions, the time when the material was in the focus of the user, and the duration of the attention.

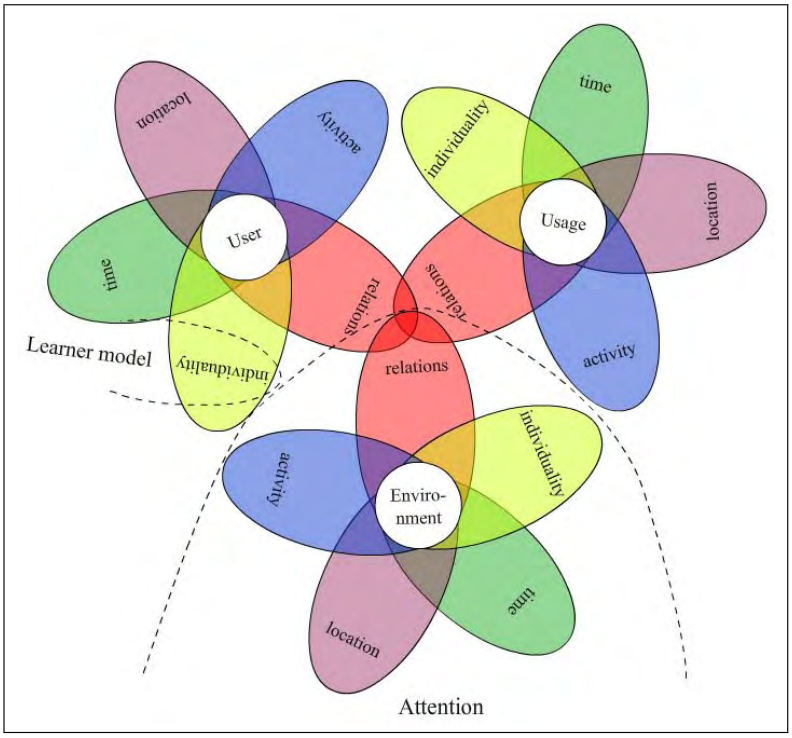


Figure 3.11: Our vision about context.

The three dimensions are linked together through the relation context, making it possible to navigate from an entity to another: starting from the user context, it is possible to identify *which* learning content has been in the focus of the user (through the environment context), but also *how*, *when*, and *how long* (through the usage context). In the rest of the manuscript, this understanding about context is used, wether terms such as context model, context (meta)data, or simply context are employed.

To handle context data described above, context-aware systems have to conform with a series of design principles both at the modeling and architectural levels [BVV+10]. The next section presents some requirements that are crucial when designing such a system.

3.2 Design Principles of Context-aware Frameworks

3.2.1 Design Principles of Context Models

A conceptual view of context-aware modeling is depicted in Figure 3.12. Input data about context is processed in an upward inference step, or so-called model acquisition, and stored within the context model [Hec05]; this information is called *raw data*. The optional downward inference step, or so-called model application, calculates new hypotheses about the context; these are called *inferred data*.

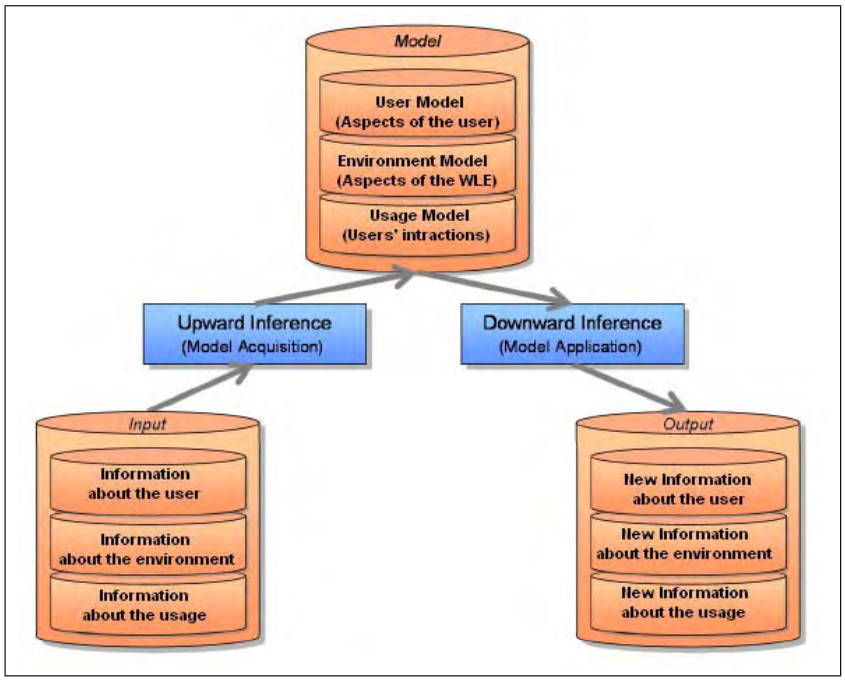


Figure 3.12: Conceptual view of context-aware modeling, inspired by [Hec05].

3.2.1.1 Raw Model

In our studies, the purpose of the context models is to capture observational data of users interacting with adaptive web-based learning environments. However, a student uses various applications to learn, for example a LMS to accomplish a quiz, and a web browser to search for extra information about that quiz. Moreover, he may use social web tools such as forums or chats to talk with colleagues or teachers. Thus, it is important for the models to be able to represent contextual data specific to heterogeneous applications. From that point, we established a set of computational requirements related to the raw model:

- **Type of formalism.** Various data structures are used for representing and exchanging contextual information from the simplest to more complex [SLP04]: key-value models, mark-up scheme

models, graphical models, object-oriented models, logic-based models, or ontology-based models;

- **User profile.** As already mentioned, a prerequisite for developing personalized services is to rely on user profiles representing users information needs. The questions that arise are related to how the profile is created and maintained: gathered from the observed applications, inferred based on collected information, or both;
- **Model flexibility.** The raw model should express different contexts: application-bounded when focuses on a single application, domain-bounded when it substantially focuses on a set of applications belonging to a specific domain, or fully general when it deals with different domains or applications;
- **Expressiveness.** The model should represent the characteristics of the raw data at different levels of details;
- **Valid context constraints.** The raw model should offer the possibility to reduce the number of admissible contexts by imposing semantic constraints (e.g. specific format for a date, a range of possible values for a property, etc.) but also semantical relation constraints between context components;
- **Extensibility.** This requirement refers to the ability of the model to be extended (on the fly) in order to take into account additional context.

On the basis of raw data, is built the inferred data presented in Chapter 2; this information is of most importance for designing and implementing powerful AdWLEs.

3.2.1.2 Inferred Model

Important criteria to be considered by context-aware systems regarding the inferred data are:

- **Integration into the context model.** Inferred data is often calculated within the client applications where it is used. Since the way it is calculated and its value are enclosed within the application, other applications cannot reuse it, instead they have to compute the data again. In order to promote the share and reuse of inferred data, it has to be modeled within the context as well;
- **Metadata to describe inferred data.** To facilitate share and reuse of inferred data, it is crucial to expose information such as the way and date it has been calculated, the type/unit of the data, etc.;
- **Persistent storage.** The value of the inferred data should be stored in a persistent way, so that multiple applications can reuse it. This is useful for inferred data which does not change (or

little) over time, but also to make possible the calculation of complex inferred data on the basis of simpler ones;

- **Up-to-date values.** A mechanism to keep the inferred data values up-to-date should be provided. Since the inferred data are calculated on the basis of raw data that quickly evolves over time, a fresh value of the inferred data should be maintained.

3.2.2 Design Principles of Context-aware Architectures

3.2.2.1 Context-aware Architectures

Context-aware systems can be implemented in many ways, depending on specific properties and requirements. [CPFJ04] identified three different architectures to design context-aware systems:

- **Built-in** systems embed context-gathering and context-processing agents, there is no clear distinctions between components that serve context modeling and components that perform other tasks;
- **Centralized.** This infrastructure introduces a layered architecture (including a storage component) that separates the collecting and processing phases. Thus heterogeneous sensors can provide contextual data to a single local storage component;
- **Distributed.** This approach extends the centralized architecture by offering remote access to the storage component, thus allowing multiple clients to remotely retrieve context data.

The distributed architecture seams to be the best design decision for building adaptive WLE, since it provides the following advantages [Kob01a]:

- Context data (e.g. user context, environment context and usage context) is stored into a central repository that can be easily queried by multiple applications at the same time;
- Context data gathered from a given application can be reused by other applications;
- The collected context is not built on the basis of a single application. Relevance of both user profiling techniques and adaptation process is thus increased;
- A unified view of data provided by different applications makes it easier to get a global view on users' behavior.

Moreover, structuring the architecture of context-aware systems into multiple layers facilitates their implementation and maintenance, since context-aware systems are based on a set of reusable blocks.

3.2.2.2 Six-layered Architecture

[HI06] suggests one of the most comprehensive architectural approaches, and introduces six layers illustrated on Figure 3.13 to design a context-aware system:

- The **gathering layer** acquires context information from sensors and then process this information to map the raw data to the raw model;
- The **reception layer** provides an interface between the gathering and the storage layers to translate data of the gathering layer to the format specific to the storage layer;
- The **storage/management layer** is responsible for storing context models and their instantiations into a repository, and ensures their consistency;
- The **query layer** provides applications with a convenient interface to retrieve the context data;
- The **adaptation layer** encapsulates the adaptation logic for the application layer;
- The **application layer** is composed of the context-aware applications which exploit the context data in order to self-adapt to the user. It is common that the adaptation and application layers are designed together.

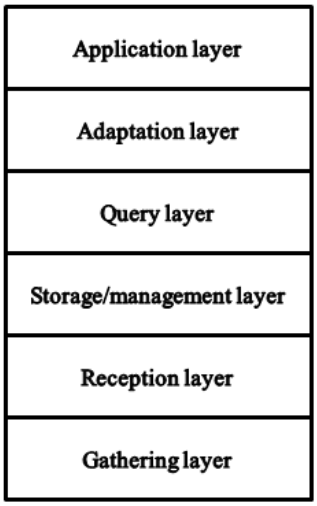


Figure 3.13: The six-layered architecture [HI06].

Starting from these outcomes, we suggest that context-aware systems meet the following architectural requirements:

- **Open framework.** The framework has to be open to easily integrate new collecting sensors and adaptive components. Being standard-aware (especially regarding the reception and query layers) as much as possible would strengthen these integrations;

- **A distributed and layered approach.** The distribution and decomposition of a context-aware system should promote both the heterogeneity of data enclosed into the repository and the easy extension of features it can provide;
- **Scalable framework.** As new context data is captured and new applications are observed, the amount of available information grows. Thus, scalable frameworks are needed.

Another hot topic that should be considered when designing context-aware systems is the user privacy, since they collect, process and store confidential and sensitive data about users.

3.2.3 Design Principles of Privacy

The largest survey ever conducted regarding citizens behaviors and attitudes concerning identity management, data protection and privacy on Internet in Europe denotes the worry of European people concerning the manipulation of their private data. The report was published in the SPECIAL EUROBAROMETER, number 359, on June 2011 [Com11]. The main findings of the survey are the following ones:

- 43% of Internet users say they have been asked for more personal information than necessary;
- A majority of European citizens are concerned about the recording of their behavior regarding credit cards (54% vs. 38%), mobile phones (49% vs. 43%) or mobile Internet (40% vs. 35%);
- To protect their identity in daily life, 62% of people give the minimum required information;
- Over half of Internet users (54%) are informed about the data collection conditions and the further uses of their data when joining a social network or registering for an online service;
- Just over a quarter of social network users (26%) and even fewer online shoppers (18%) feel in complete control;
- 70% of European citizens consider that their personal data held by companies may be used for a purpose other than that for which it was collected.

As stated above, users are much concerned about how their personal data is used on Internet. A directive of the European Commision [Com95] defines personal data as *any information relating to an identified or identifiable natural person ('data subject'); an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity*.

To protect individuals from the consequences of this knowledge, laws regulating the type of protection that personal data should receive, and rights that individuals enjoy under their data, emerged.

Thus, seven principles of privacy identified by various commissions (OECD's[1] recommendation for protection of personal data, directives 95/46 and 2002/58 of the European Commission) about the protection of personal data should be addressed by e-learning systems:

- **Notice.** Data subjects should be informed that their personal data is being collected;
- **Purpose.** Data must be collected for a specific purpose and not for any other purposes;
- **Consent.** Subjects should consent for collection of personal data;
- **Security.** An organisation that uses personal data must implement appropriate techniques to protect data against destruction, accidental loss or unauthorized access;
- **Disclosure.** Data should not be disclosed to third parties without the consent of the subject;
- **Access.** Data subjects should be allowed to access their data and make corrections to any inaccurate data;
- **Accountability.** Data subjects should be able to hold data collectors accountable for checking the above principles.

The mechanisms required to ensure the above principles of privacy engage a series of laborious and expensive resources. The anonymous and pseudo-anonymous access to a personalized system represent an alternative, since it is not subject to privacy laws anymore [Kob01b]. [FM95] identified six levels of anonymity that vary from the univocal association of data to a person, to the total disengagement of the data of a person:

- **Super-identification.** The identity of a person is attributed to an external administrative entity that is located outside the adaptive system (e.g. X.509);
- **Identification.** The user identifies himself and demonstrates knowledge of a secret (e.g. password) that is compared by the system with a stored value (e.g. Unix authentication);
- **Latent identification.** The user identifies himself in the system and adopts one of the predefined set of pseudonyms;
- **Pseudonymous identification.** The user generates a pseudonym and a secret (e.g. password) that will be used for each session. The system is unable to reveal the identity of a person;
- **Anonymous identification.** The user gains access to a system by providing a secret (e.g. password) without disclosing his identity. The system is unable to distinguish among users who have knowledge of the same secret;

[1] The Organisation for Economic Co-operation and Development

- **Anonymity.** The user is not identified nor authenticated in the system. The system is unable to distinguish between different users even if it handles context data (e.g. visits of virtual museums).

With super-identification, identification and latent identification, the adaptive system is able to identify users and distinct sessions of the same user. The drawback is that the user can be identified, thus privacy concerns must be addressed.

The anonymous identification associates a user model with a secret, that is a group of persons which uses the same secret to authenticate. Even if the privacy concerns are not applicable (the system is unable to distinguish among users which use the same secret), the group adaptation is not of most importance for an AdWLE.

With anonymity, the system cannot reveal the identity of the user, thus no privacy issues have to be addressed by the adaptive application. Nevertheless, the application can't distinguish among users, at best it is able to distinguish between different sessions (but it cannot link sessions); but the user modeling performed at the session level makes adaptation very poor.

The pseudonymous identification seems to be the best compromise between detailed context models and respect of privacy principles: it is possible to differentiate users based on their pseudonym without disclosing their identity, and successive sessions can be interconnected to make the long-term modeling possible.

After this review of design principles related to context-aware frameworks, the next section describes and analyses some systems to see how they take into account these principles.

3.3 Approaches Dealing with Context

3.3.1 TaskTracer

TaskTracer [DDJ+05] is based on a key-value model to help multitasking knowledge workers rapidly locate, discover and reuse past processes to successfully complete tasks. TaskTracer collects environment and usage contexts from observations of user activities when interactions occur with desktop applications. The data is collected as user interface event messages (UI event), each event being composed of various information (event type, content of resources, timestamp, etc.).

At the initial step of data collection, the user is invited to manually specify what task he/she is doing, so that each UI event is associated to that current task. After enough data collection, the user is not asked anymore to precise the task he/she is working on, the system is able to predict the current task and thus to restore all applications that were previously involved in that task.

TaskTracer adopts a central architecture illustrated on Figure 3.14. It is based on the Publisher-Subscriber paradigm to separate the context collecting and processing phases. To collect context data,

TaskTracer uses a COM[1] plugin attached to Microsoft™ Office applications, a Windows CBT[2] hook, a .NET FileSystem Clipboard class, a hook to the Windows Clipboard and a hook to a phone modem. The sensed data is received by the Publisher (reception layer) and then stored into a database relying on the SQL query language; Subscriber applications then access the data for further processing.

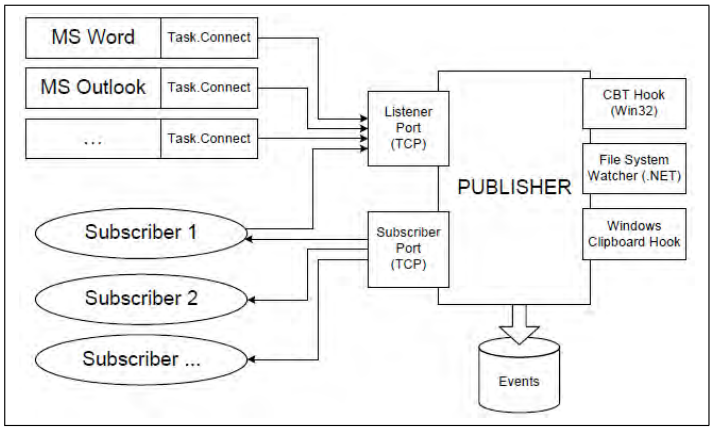


Figure 3.14: TaskTracer publisher-subscriber architecture [DDJ+05].

3.3.2 CAM Framework

Attention metadata, or Attention.XML, is an open specification for tracking, prioritizing and sharing people's attention (e.g. what people are reading, looking at or listening to), and was introduced into the field of information technology by Steve Gillmore, the president of AttentionTrust[3]. The design of Attention.XML is based on three premises: (1) attention flows are recorders for single users, (2) attention records are bags of data objects that have been in the users' focus, and (3) users receive data objects through diverse channels and the objects are stored according to these channels.

The Attention.XML schema is described on the left-hand side of Figure 3.15. The root element, *group*, comprises a *title* (e.g. the name of the user the attention is collected for) and a *feed* element which describes the set of channels (blog, web site, etc.) providing data object. The feed element is described by several elements (e.g. *title, url, lastread, read times*, etc.), and includes an *item* which represents the objects that were in the focus of the user.

The missing usage context was considered as a major drawback of this schema. Therefore, [WNVD07] introduced the Contextualized Attention Metadata (CAM) schema and architecture as an extension of Attention.XML to capture behavioral information of users.

As shown on Figure 3.15, CAM focuses on *event* describing the action that occurs on data objects. Events are described with a timestamp and a description, among others. An event can be associated to

[1]Component Object Model
[2]Computer-Based Training
[3]AttentionTrust - http://www.attentiontrust.org/

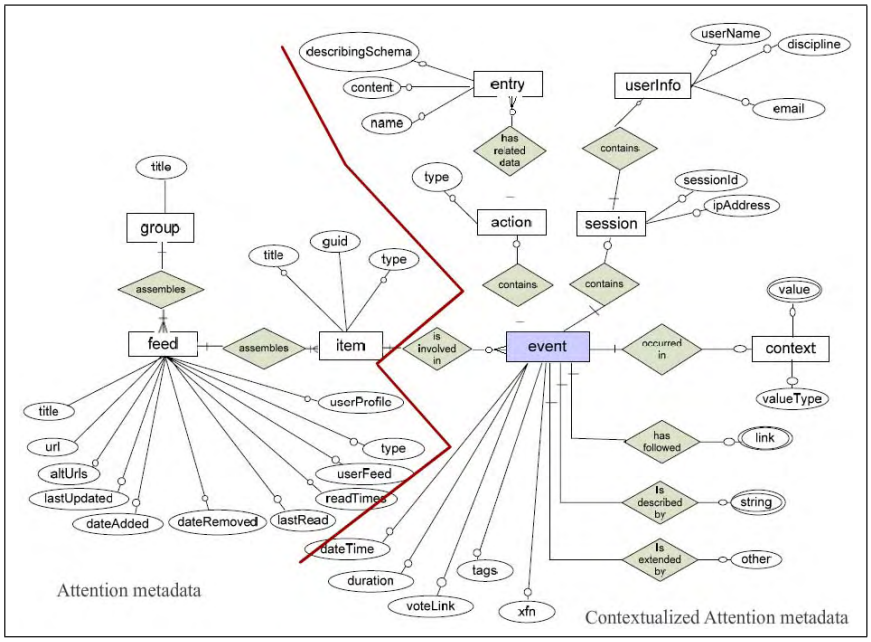


Figure 3.15: CAM schema [WNVD07].

an *action* of a certain *type* and detailed by *related data*. Further, events occur in a certain *context* and are part of a technical *session*.

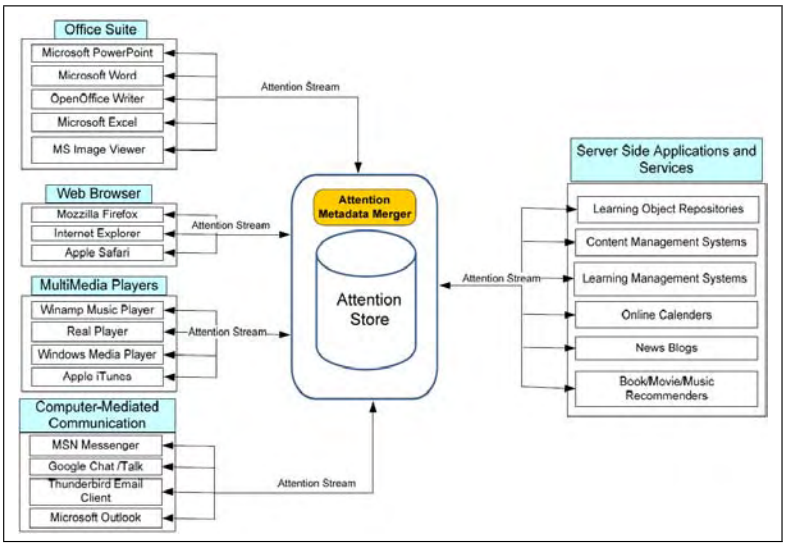


Figure 3.16: CAM architecture [WNVD07].

The CAM architecture illustrated on Figure 3.16 allows collecting attention metadata from any desktop or server side application, merges data into a single stream per user, and stores data into the Attention store. The gathering layer of the current CAM implementation comprises various tools and addons to collect attention metadata: the Ariadne Finder [TVP+09] and the MACE project [SDVC+07] use

integrated sensors to collect CAM, whereas the CAMera framework [SFJ+09] uses add-ons for Thunderbird, Skype, Firefox, MS Outlook, MS Power Point, MS Word and the Flash meeting system. The reception layer is implemented by the Simple Publishing Interface (SPI) [TMVA+08], whereas XML-based repositories represent the storage layer. The Simple Query Interface (SQI) [SMVA+05] is finally used as the query layer.

3.3.3 WildCAT

WildCAT [DL05] is an extensible Java framework which aims at easying the creation of context-aware applications where heterogeneous information can be shared and reused. It contains a generic context model schema (see Figure 3.17) that supports different levels of extensions, from the simple configuration of the default generic implementation, to completely new implementations tailored to specific needs. The context information can be accessed through two complementary interfaces: synchronous requests and asynchronous notifications.

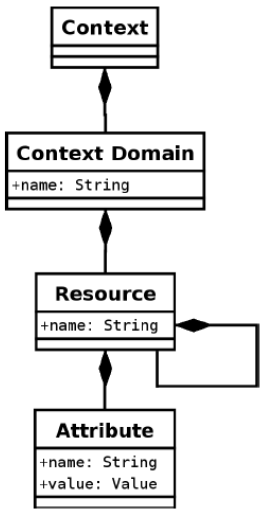


Figure 3.17: WildCAT context model [DL05].

The generic model contains as super class the *Context* class. The context is made of several domains, each represented by a *ContextDomain* object (e.g. system, applications, users, etc.). The purpose of context domains is to separate the different aspects of the context, and to allow each of these to use a custom implementation. The context domain is modeled as a tree of resources, each being described by attributes (simple key/value pairs).

3.3.4 KnowledgeTree

KnowledgeTree [BSL+08] is an architecture for adaptive e-learning, and is based on distributed and reusable intelligent learning activities. Its architecture comprises four kinds of servers depicted on Figure 3.18:

- A **portal** is similar to a LMS. It provides a centralized single-login point for enrolled students to interact with all tools and learning contents provided in the context of their courses;
- An **activity server** plays the similar role as a LOR, containing highly adaptive and reusable learning materials;
- A **value-adding service** adds some valuable features such as adaptive sequencing, annotation, visualization or content integration to the activity servers;
- The **student model server** represents the needs of students. It collects student performance from each portal and activity servers, and forwards this information to adaptive portals and activity servers.

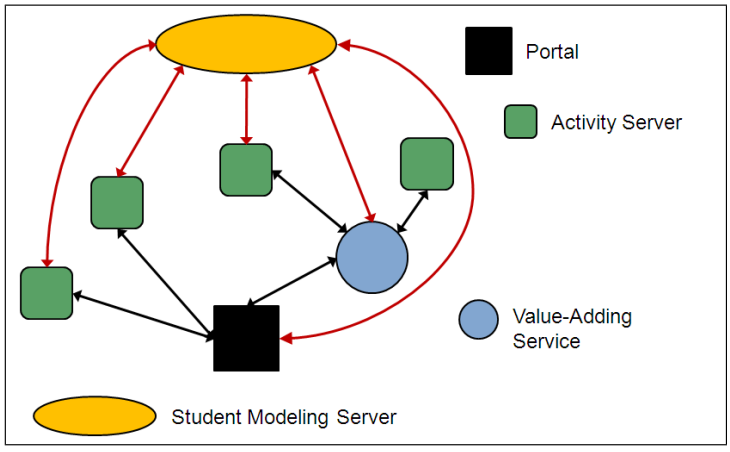


Figure 3.18: KnowledgeTree distributed architecture [BSL+08].

The framework collects events (e.g. page is read, question is answered, etc.) provided by the various servers, and stores them within the event storage part of the student model. This last one contains two storages: one for collected events and another for inferred properties. Indeed, the flow of events is further processed by inference agents that update the values of inferred properties (i.e. current motivation level, current level of knowledge, etc.).

3.3.5 Dyonipos

Dyonipos [RDL09] is a Personal Information Management (PIM) application that automatically identifies the work task of the user and then provides him/her with information from both personal and organizational environments. The context (see Figure 3.19) is seen as a semantic pyramid implemented through an ontology composed of five dimensions: (1) the action dimension models user actions, (2) the resource dimension refers to resources of the desktop computer, (3) the user dimension integrates concepts about the user, (4) the application dimension is a property of the *Event* concept, and (5) the information need dimension represents the pro-active information.

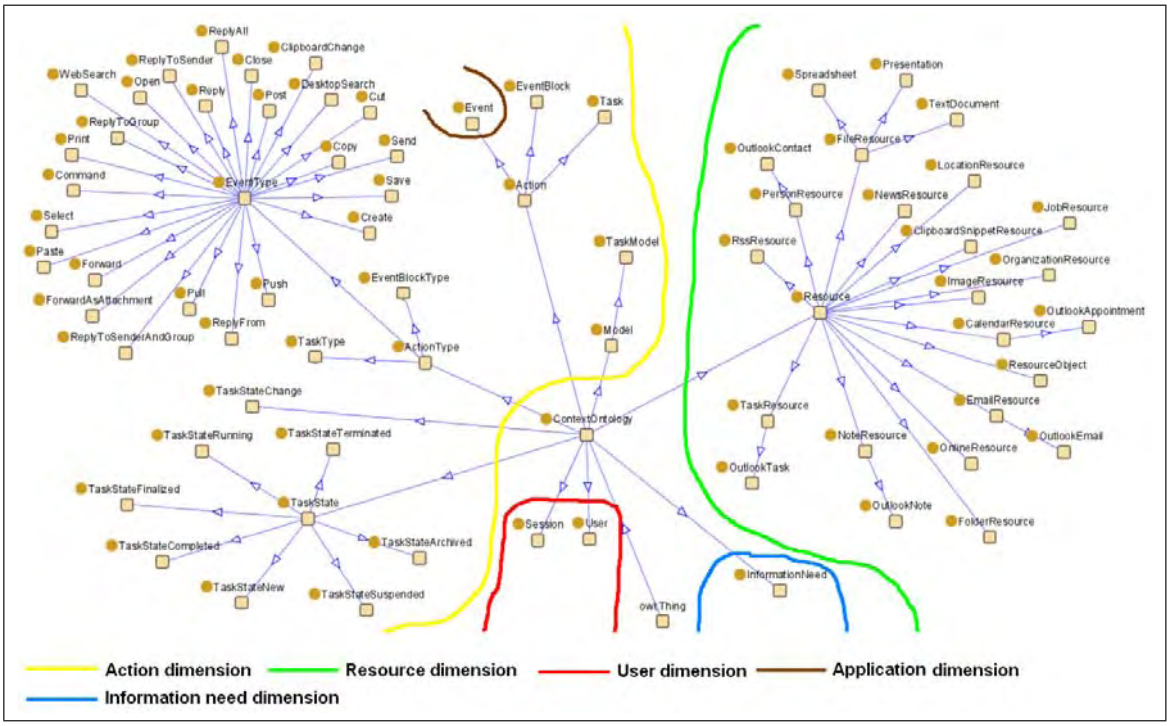


Figure 3.19: User interaction context ontology [RDL09].

The architecture of Dyonipos is illustrated on Figure 3.20. Context Observers are programs, macros or plug-ins which collect observational data from desktop applications; these sensors send usage data as event stream to the reception layer represented by the Dyonipos Task Recognizer. This last one is responsible for detecting the current information needs of the user, and then identifying resources that are of relevance for the user. To achieve these tasks, the Task Recognizer communicates (through the m2n components) with the KnowMiner framework, a service-oriented knowledge discovery framework. It provides access to indexing, search, information extraction, clustering and classification services both on the client and server sides; the server side is also responsible for the federation of the context models of all users further exploited by the Task and Process Mining components to process adaptation on a multiple user scale. Finally, the Dyonipos Task Recognizer GUI offers the user interface.

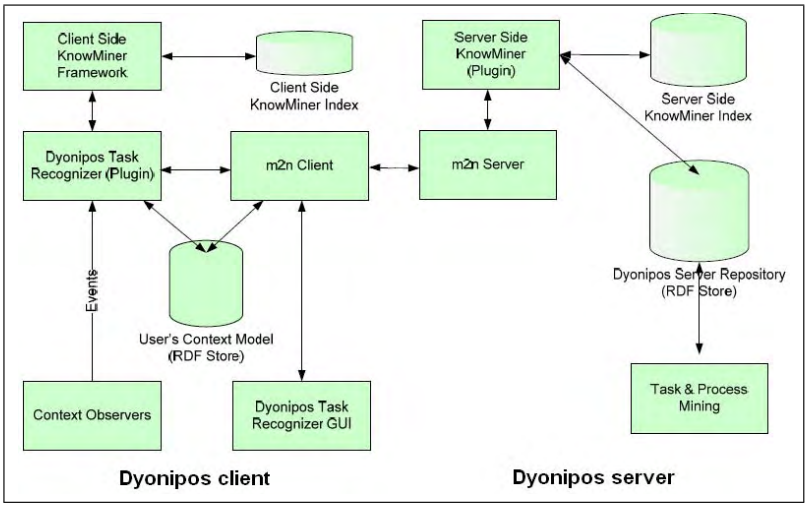


Figure 3.20: Architecture of Dyonipos [RKL+08].

3.3.6 Aposdle

APOSDLE [LM06] is a knowledge work support system which aims at enhancing the productivity of workers by integrating learning within everyday work tasks.

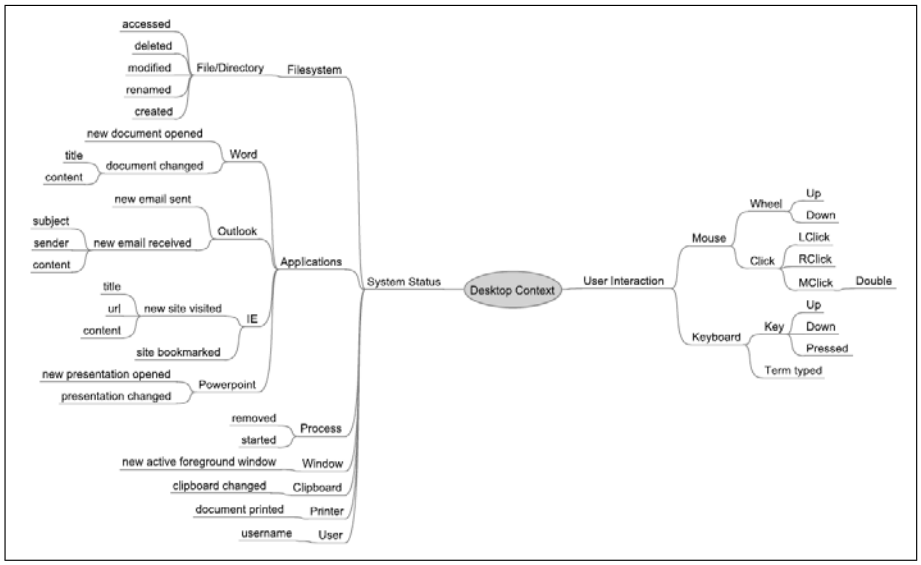


Figure 3.21: APOSDLE context [LFBG07].

The architecture of APOSDLE is presented in Figure 3.22. During the work process, the APOSDLE context sensors log any desktop events reflecting users actions (see Figure 3.21 for an overview of contextual information). From these raw context events, the current task of the user is inferred. This task is logged by the *work context logging service* and stored into the user model. The activities specific to resources are also logged and stored into the user model through the *resource activity logging service*.

The history of tasks executions and all resource-based actions are delivered by the *usage data history service* to provide an overview of how learning goals evolve over time. The *usage data control service* gives users the opportunity to delete or modify their usage data.

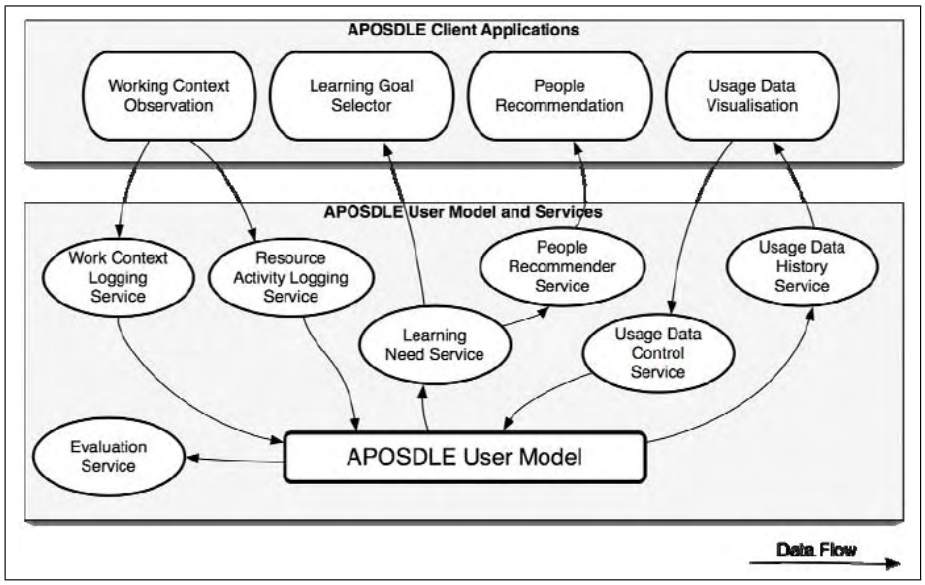


Figure 3.22: APOSDLE architecture [LBKL09].

The knowledge and skills required to successfully perform the detected task are compared with those of the user model, so that a learning need can be identified by the *learning need service*. In order to help users to achieve goals, the *people recommender service* aims at finding people within the organization which have expertise related to the current goal.

3.3.7 Usage Tracking Language

Usage Tracking Language (UTL) *is designed as a generic language to describe tracks and their semantics, including the definition of the observation needs and the means required for data acquisition* [CI06].

UTL structures tracks according to two types of data (see Figure 3.23): primary and derived. Primary data includes raw data (events issued from the learning environment), content data (data produced by learners) and additional data (tutors' annotations, pedagogical scenario, etc.). Derived data, computed based on primary or other derived data, differs depending on the data it represents: intermediary data from a calculus, or indicators.

While UTL allows to describe the information needed to formalize an indicator, the description of the data acquisition method is informal, being difficult to automatically generate analysis tools to compute indicators [NIC10]. To solve this problem, [NICK09] proposed an extension part of UTL, called Data Combination Language for UTL (DCL4UTL). It comprises a formal grammar for calculating indi-

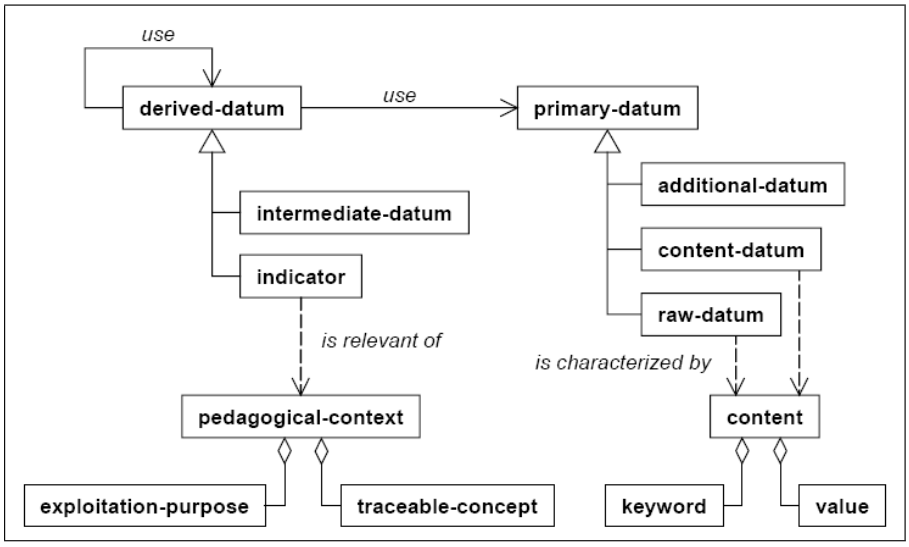


Figure 3.23: Conceptual model of a track in UTL [CI07].

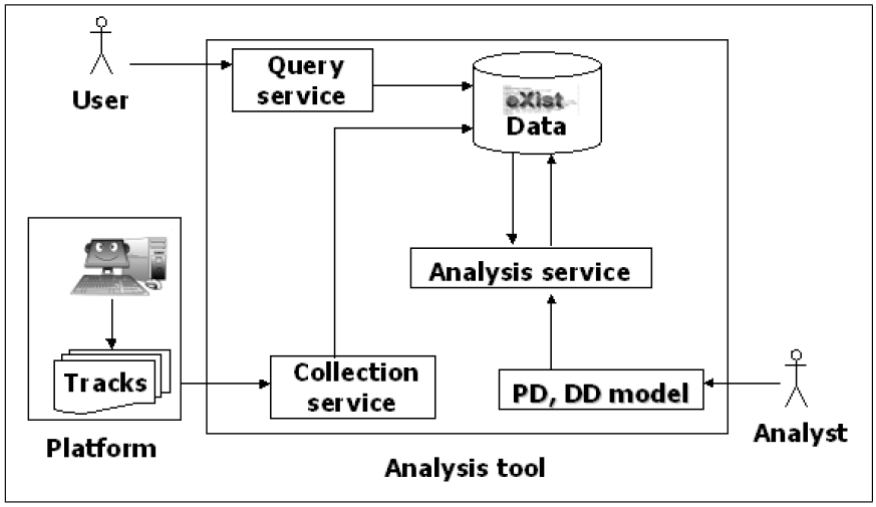


Figure 3.24: Architecture of the analysis tool in UTL [NICK09].

cators by reusing the concepts proposed by UTL. The indicator model features three facets: *Defining* defines the name and description of the indicator, *Getting* describes the means for calculating the value of the indicator, and *Using* defines the pedagogical use and purpose of the indicator. The architecture of UTL implementation and indicators specification appears in Figure 3.24:

- **The collection service** collects the context from the learning platform and stores the matching XML data into an eXist[1] database;
- **The interface module** helps analysts to build and define some indicator calculation methods. It also helps teachers to exploit the calculation results;

[1] http://exist-db.org/

- **The analysis service** interprets the calculation methods, calculates the indicator and formats the result according to the specified format;

- **The query service** allows users to query for context data.

3.3.8 TBS-IM

[DSP+10] presents a theoretical and practical framework to calculate activity indicators within a web-based learning environment. The system collects raw data from various tracking sources available within the learning system, and applies a tracking model over this data to obtain the so called *trace premiere*, or primary trace. Furthermore, a sequence of transformations over the primary trace is applied in order to produce more specific traces (intermediary traces), and finally the indicators.

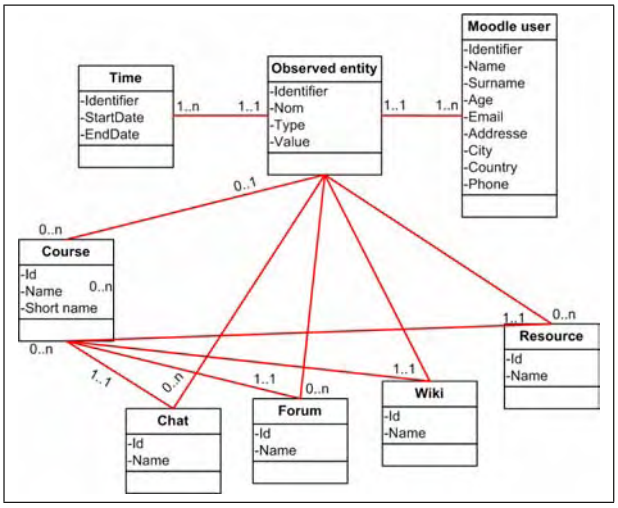


Figure 3.25: The content of the primary trace in TBS-IM [DSP+10].

An implementation of this framework was conducted for Moodle, named Trace-Based System for calculating Indicators in Moodle (TBS-IM). The architecture of the framework is depicted in Figure 3.26 and is composed of three modules:

- **The collecting module** generates the primary trace based on Moodle context sources, and is characterized by both a server and a client side components. The server side module generates the primary trace in HTML to be visualized, but also OWL to be further reused. The client side module allows users to perform a parameterized collect (i.e. based on classes and attributes) and to generate primary traces relevant to their needs using the filtering module. The primary trace is generated in OWL for further reuse, but in the internal database format as well in order to act as input of the transformation module (see Figure 3.25);

- **The transformation module** uses operators to transform the primary trace into an intermediary trace. Intermediary traces are stored into the database as well, thus facilitating the process of building similar traces;
- **The indicator calculation module** associates to each indicator one or more intermediary traces and a calculation rule. To calculate the indicator, intermediary traces are loaded by this module, statistical data is computed over the traces (see Figure 3.25) and integrated into the calculation rule which is finally processed.

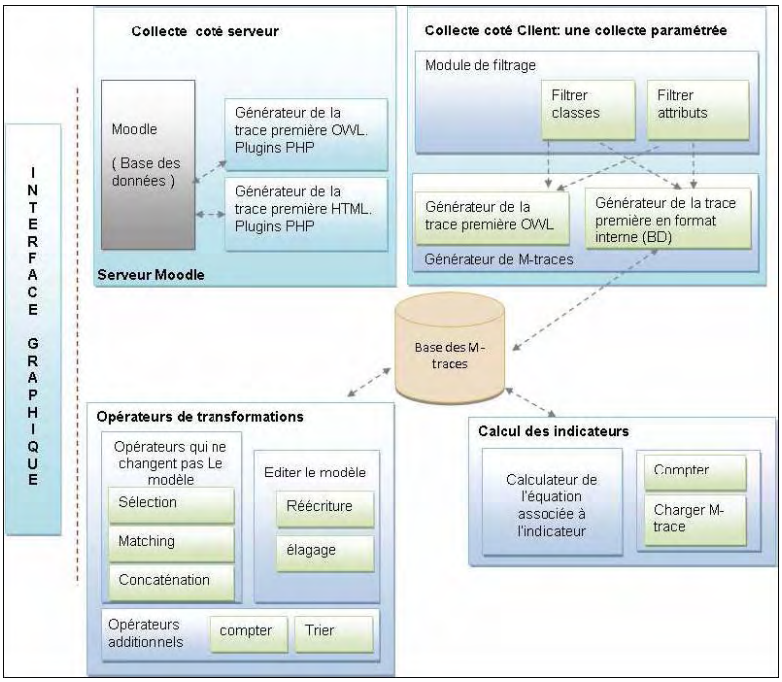


Figure 3.26: Architecture of SBT-IM [DSP+10].

3.3.9 Learning Registry

The Learning Registry [BBK12] is an infrastructure that enables instructors, teachers, trainees and students to discover and use the learning resources held by various American federal agencies and international partners. Learning Registry stores more than traditional descriptive data (metadata) for a learning resource, including social data such as tags, comments or ratings as well. These data, also called *paradata*, are further shared in a common pool for aggregation, amplification and analysis.

The paradata schema illustrated on Figure 3.27 is very similar to the CAM schema. The root element is the collection element which contains as sub-elements activity items. Each activity is further characterized by (1) an actor describing the entity or person that performed the activity, (2) a verb translating the type of performed activity, (3) the object that the action is applied to, (4) a list of objects related

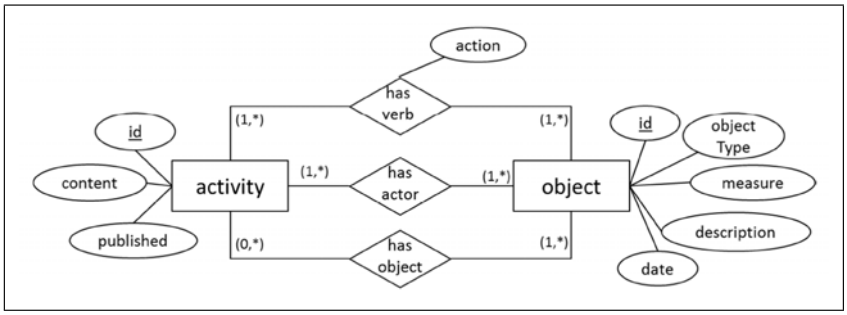


Figure 3.27: Simplified Learning Registry paradata schema [NSW12].

to the current object, and (5) a content element comprising the id of the activity, a natural language description of it and the timestamp when the activity was published.

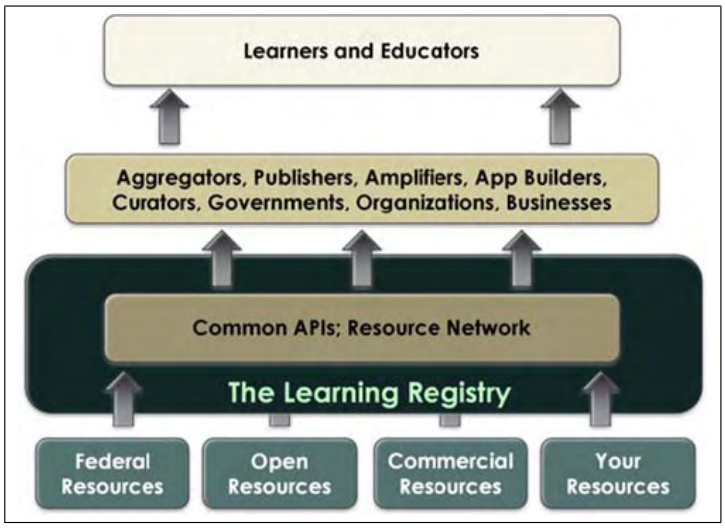


Figure 3.28: Architecture of Learning Registry [JR11].

The architecture of Learning Registry is divided into four layers (see Figure 3.28):

1. At the bottom level are the learning resources, which are external to the Learning Registry Network;

2. The second level, the core of this architecture, is a distributed network of nodes holding a collection of resource descriptions and offering a set of services: replication, publication and search;

3. The third level represents the client applications that reuse the network of resources;

4. At the top level are the communities of learners and educators that benefit from these applications and tools.

3.3.10 NSDL Paradata

The National Science Digital Library (NSDL)[1] is a provider of quality digital resources to science, technology, engineering and mathematics education communities. The resources are described by their metadata (in terms of LOM data), but also by usage data and paradata [Blo12] (see Figure 3.29).

Each NSDL record is uniquely identified and contains at least the URL of the resource to which the paradata applies. Other information describes a record as well, since any additional elements can be specified into the schema. The most important element is the *usageDataSummary* which comprises all available usage statistics/information about a resource. The schema defines five different value types: (1) *integer/float* represents the number of times certain actions have been performed on the resource (e.g. awarded, cited, commented, downloaded, tagged, viewed); (2) *string* is a textual value that has been associated to the resource (commented, tagged); (3) *rating type* is the numeric average that evaluates resources (star, usability); (4) *vote type* represents the number of positive and negative responses to a resource (accurate, like, useful); (5) *rank type* represents the standing of a resource in a hierarchy (best, most).

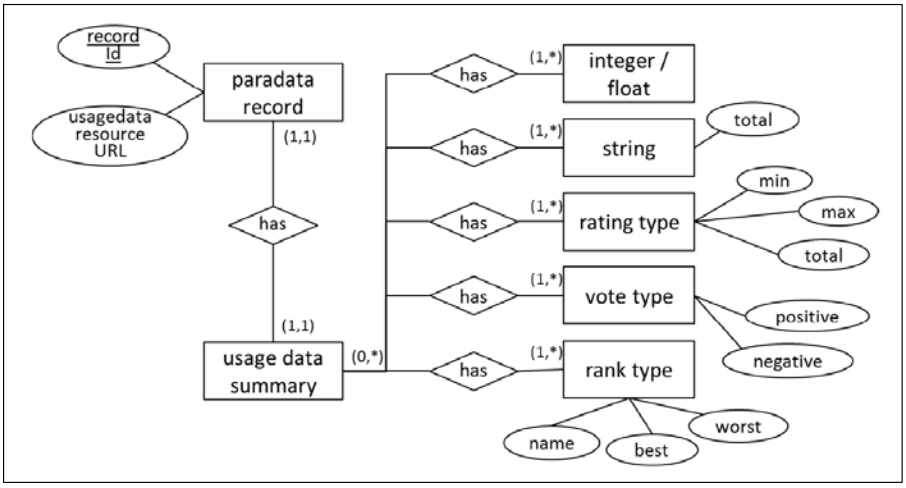


Figure 3.29: Simplified NSDL paradata schema [NSW12].

Besides its type and value, each *usageDataSummary* element contains the begin and end date for the usage data (*dateTimeStart*, *endTimeStart*), information about the *audience* that conducted the event (educator, student, general public), the *subject* of the used resource (computing, engineering, mathematics, science, technology), and the educational level (*EdLevel*) the resource was used.

From the architectural point of view, NSDL is composed of four layers depicted in Figure 3.30:

- The **ingest layer** is composed of two modules: the OAI-PMH module harvests collections of metadata (including paradata) from partners to inject them into the repository, and the NSDL Collec-

[1] http://www.nsdl.org

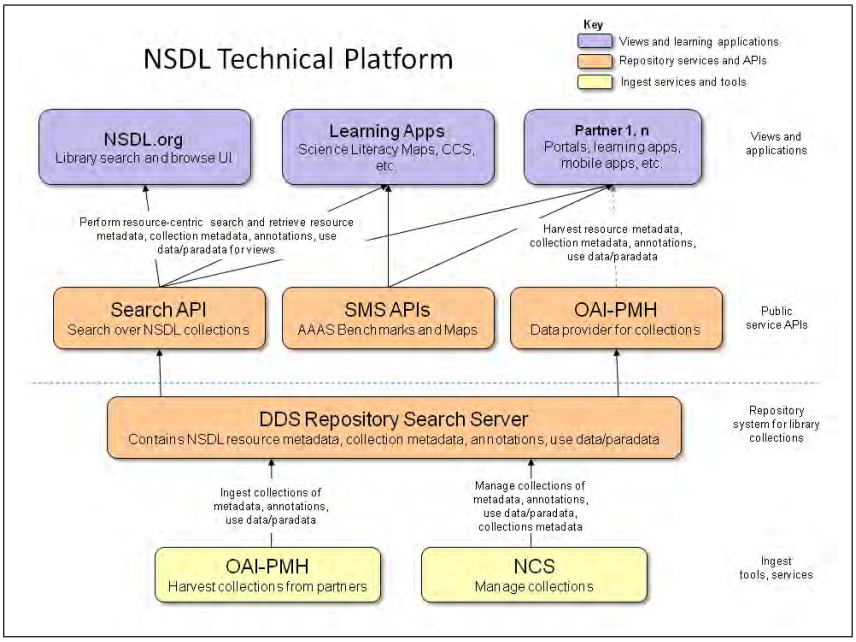


Figure 3.30: NSDL architecture [Wea12].

tion System (NCS) manages the collections of metadata, paradata and resources;

- The **repository layer** stores both metadata and paradata in XML format, and provides a search API built on top of Lucene[1];
- The **public service APIs layer** provides three APIs to search and harvest metadata and paradata, and to help teachers and students better understand the relationships between science concepts;
- The **views and applications layer** comprises the applications that use data stored into the DDS Repository and available through the previous layer.

3.3.11 Comparative Tables

In order to offer a clear overview of the approaches presented in this section, some comparative tables analysing the adopted models and architectures are discussed below.

3.3.11.1 Models

3.3.11.1.1 Raw Model

Table 3.1 summarizes the features of the various approaches from the raw model point of view. Due to its very simple structure, the **TaskTracer** key-value model is not expressive. Even if it may collect

[1] http://lucene.apache.org/

contextual data from any desktop application, information describing context is poor in number, and doesn't vary from one application to another, which makes TaskTracer focusing on a specific context subproblem. Semantic constraints are not applicable (data type constraints only), and the model doesn't take into account the user context.

Within the **CAM framework**, the user context is defined by the *userProfile* element which references the profile of the user within the observed application; therefore, a single user is described by as many user contexts as the number of applications he/she interacts with while learning, they are not merged together. The model shows some limits when it comes to represent context of any application, due to predefined and fixed context metadata (e.g. the *item* element is defined by three properties, and it is not possible to specify additional attributes). However, elements characterized by an unlimited multiplicity (e.g. the *entry* element) allow defining new context metadata, but the absence of relationship and granularity levels between context components prevent CAM to offer an expressive model.

System / Model criteria	Context: User	Context: Environment	Context: Usage	Formalism: Key - value	Formalism: Mark - up	Formalism: Object	Formalism: Logic	Formalism: Graph	Formalism: Ontology	Flexibility	Expresivness	Context constraints	Extensibility
Tasktracer		✓	✓	✓						A	L	M	L
CAM	✓	✓	✓		✓					D	M	L	M
WildCAT						✓				G	H	M	H
KnowledgeTree	✓	✓	✓	✓			✓			A	L	M	L
Dyonipos	✓	✓	✓				✓		✓	D	H	H	H
Aposdle	✓	✓	✓	✓			✓			D	L	M	L
UTL	✓	✓	✓		✓					A	M	L	H
TBS-IM	✓	✓	✓						✓	A	H	H	L
Learning Registry	✓	✓	✓			✓				D	M	L	H
NSDL paradata		✓	✓		✓					D	M	L	M

A: Application-bounded, D: Domain-bounded, G: General

L: Low, M: Medium, H: High

Table 3.1: Raw model comparative table.

In **WildCAT**, the different types of context are not explicitly defined, but they can be modeled through extension API. The model is thus highly extensible, thanks to its abstract level of representation. Furthermore, composition relationships make it possible to define semantic constraints between context entities.

KnowledgeTree supports only a specific context domain (e-learning), and tackles the problem of reusing intelligent educational learning activities only. The model of context entities is divided into two levels: the lower level represents events, and on its basis, the upper level describing the user model is inferred. Both lower and upper contexts miss granularity, and no relationships between entities are defined. Only the upper context metadata can be extended, since new properties can be inferred us-

ing additional inference agents. Because of its key-value representation, semantic constraints are not applicable (data type constraints only).

Compared to CAM, the model of **Dyonipos** can be easily extended with concepts and properties, but also with relationships between resources (at various granularity levels); these features thus provide a high expressiveness and context validation.

The raw model in **APOSDLE** is represented by two types of data: the tasks performed by users and the events sensed by agents from various desktop applications. APOSDLE is similar to KnowledgeTree in modeling context at two different levels of abstraction, but both lower and upper models are rather simple (i.e. key-value pairs) and don't describe context at different levels of details. While the low context events are hardly extensible, new higher level context (i.e. indicators) may be easily deduced.

The **UTL** model integrates all context dimensions, and is represented by relations between the resource (i.e. traceable concept) for which data is collected, the usage of the observation (e.g. managing), and the activities performed over the resources; relationships between entities can be defined as well. The raw model can be easily extended by adding new entities, but in the same time lacks semantic constraints. Finally UTL is specific to e-learning, and more precisely to learning scenarios.

The model of **TBS-IM** is application-bounded but quite expressive, since relations between various entities are described (see Figure 3.25). The context constraints (semantics and types) are ensured by the internal database of the target application, but the raw model is not extensible since the tables recording data about users are predefined.

In **Learning Registry**, the context model is specific to a domain of applications, being in the same time highly extensible since no restrictions are imposed on adding new properties to any element of the schema. However it is not much expressive, and the lack of semantic may introduce difficulty in exploiting the stored information.

The **NSDL paradata** schema is very similar to the Learning Registry model, but the user is missing. The model is domain-bounded, allowing to collect context from various applications with similar characteristics, and not much expressive, being characterized by a minimum set of elements. Some of them lack semantics and may introduce redundancy. NSDL paradata schema is similar to CAM regarding model extensibility: elements characterized by an unlimited multiplicity allow defining new context metadata.

3.3.11.1.2 Inferred Model

If the previous section tries to analyse the existing approaches according to the context data directly collected from the observed applications, we focus here on the inferred information that can be calculated on the basis of the raw context; Table 3.2 summarizes our study from this point of view.

TaskTracer does not offer any mean to represent indicators, it identifies the current task of the user from the raw context only. The algorithm and the value of this indicator are not reusable, even if the system maintains an up-to-date value of the indicator.

System / Model criteria	Metadata	Design	Algorithm reuse	Value reuse	Value up-to-date	Algorithm	Complexity	Heterogeneous context
TaskTracer	✗	Difficult	✗	✗	✓	Seq. of ev. compare	+++	✗
CAM framework	✗	Difficult	✗	✗	✗	Statistical	+	✓
WildCAT								
KnowledgeTree	✓	Difficult	✗	✓	✓	Not specified	+++	✗
Dyonipos	✗	Difficult	✗	✗	✓	Keyword compare, etc.	++	✓
Aposdle	✓	Difficult	✗	✓	✓	Seq. of ev. compare	+++	✓
UTL	✓	Difficult	✗	✓	✗	Statistical, arithmetic	+	✗
TBS-IM	✓	Easy	✓	✓	✗	Statistical, arithmetic	+	✗
Learning Registry	✓	Difficult	✗	✓	✓	Statistical, arithmetic	+	✓
NSDL paradata	✓	Difficult	✗	✓	✓	Statistical	+	✓

✓: Implemented, ✗: Not implemented

+: Simple, ++: Medium, ++: Complex

Table 3.2: Inferred model comparative table.

Just like TaskTracer, the **CAM framework** does not offer any mean to represent indicators. At our knowledge, only statistical indicators have been calculated by third party applications.

WildCAT offers a framework to define context, but unfortunately we find no information about how the context is further exploited.

The indicators in **KnowledgeTree** are calculated through external and internal inference agents which process the flow of events in different ways, and update their values in the inference model. Thus, values are reusable but a new inference agent has to be created if similar indicators must be calculated (algorithms can't be reused).

Dyonipos calculates a single indicator (information need) through a dedicated component. Therefore, the value can't be reused, nor the algorithm used to process the calculation.

APOSDLE is also interested in a single indicator (the current task of the user), and stores its up-to-date value into the context repository. The value of this indicator can thus be reused, but the algorithm is not available.

Compared to the other approaches, **UTL** offers a complete information model for defining (UTL) and calculating (UTL-CL) indicators. Indeed, derived data can be defined and computed on the basis of any primary (or other derived) data collected directly from the learning environments. The drawback is that the specificity of the framework makes it difficult to reuse context data.

The approach proposed by [DSP+10] is very similar to UTL. It also proposes a sequence of transformations starting from a primary trace and resulting in any kind of indicators that can be reused inside the framework. However, values are calculated once unless it is explicitly requested.

Learning Registry and **NSDL** represent inferred data into the model, but its value is calculated outside the framework. Additional indicators can be further calculated by querying the distributed context repository.

To conclude, it is obvious that systems which interact directly with the context repositories to compute inferred data are able to generate indicators characterized by any level of complexity; third party applications or proxies are able to build their own algorithms and queries. At the opposite, even if UTL and TBS-IM provide a GUI to easily define new indicators, these frameworks are limited in terms of complexity. Indeed, the approach suggested by [DSP+10] to specify the calculation rules of an indicator is based on a set of predefined transformations applied to context data. Therefore, users are not able to define new indicators that require a mechanism that is not included into the predefined set of transformations.

3.3.11.2 Architectures

TaskTracer relates on a middleware architecture since there is a clear distinction between the collecting layer and the storage layer, events being stored within a relational database. TaskTracer complies with the publisher-subscriber paradigm to share the collected users' activity, but scalability issues may appear due to the huge number of events that can be collected. Furthermore, the data may leave the users PC and privacy concerns should be addressed; no information about this criteria has been found.

CAM's distributed architecture and its SQI web services make it possible to widely share context data with remote client applications, but security and disclosure privacy principles may not be respected: sensitive data may be retrieved by any application sending a request to the SQI search service. If new applications have to be observed, the only thing to do is to implement agents specific to the concerned applications. Finally, the XML format representation and the huge number of CAMs produced by heterogeneous applications introduce scalability issues.

The **WildCAT** framework was designed to facilitate the acquisition and aggregation of contextual data in order to offer a view of the execution context of many applications. Despite this, contextual information can be accessed only by local Java applications, thus restricting interactions with other programming languages and tools and making the process of extension very tedious (creation of additional sensor libraries and configuration files). Even if the user profile is not explicitly modeled, it can be represented as a context domain and thus may introduce privacy concerns that are not taken into account.

The **KnowledgeTree** framework is opened in the sense that events are shared with internal and external inference agents, while the inferred properties are shared to applications and tools integrated within the distributed architecture (but not accessible from applications outside the framework). Scalability issues may be avoided, thanks to the possibility of adding new student model servers to ensure load balancing. Moreover, users' privacy concerns are ensured by pseudonymous identification.

System / Architecture criteria	Arch.: Built in model	Arch.: Centralized	Arch.: Distributed	Open framework	Sharing	Scalability	Privacy: Notice	Privacy: Purpose	Privacy: Consent	Privacy: Security	Privacy: Disclosure	Privacy: Access	Privacy: Accountability
Tasktracer		✓		L	M	M	-	-	-	-	-	-	-
CAM			✓	H	H	L	✓	✓	✓	-	-	✓	✓
WildCAT		✓		M	M	L	-	-	-	-	-	-	-
KnowledgeTree			✓	M	M	H	✓	✓	✓	✓	✓	✓	✓
Dyonipos			✓	H	M	L	✓	✓	✓	-	-	✗	✓
Aposdle			✓	L	M	L	✓	✓	✓	-	-	✓	✓
UTL			✓	L	M	M	-	-	-	-	-	-	-
SBT-IM			✓	L	M	L	-	-	-	-	-	-	-
Learning Registry			✓	H	H	H	-	-	-	-	-	-	-
NSDL paradata			✓	M	H	M	✓	✓	✓	✓	✓	✓	✓

✓: Implemented, ✗: Not implemented, -: No information

L: Low, M: Medium, H: High

Table 3.3: Architecture comparative table.

Even if **Dyonipos** context information is enclosed at the organizational level, its open architecture makes it possible to integrate a wide range of collecting agents while other applications and services can be built upon it and access the encapsulated context information in a uniform way. Context information is stored both on a central server and on the PC of users, the last ones deciding what information to share on the central server; this approach thus ensures the privacy of the user. However, ontologies are heavy and slow to use (both for human users and applications), thus introducing scalability issues when the framework comes to deal with a large amount of data.

APOSDLE sensors are not built into the sensed applications, but implemented as software hooks that operate at the operating system level to continuously log the data. Thanks to the production services, the collected context is made available to other services of the APOSDLE environment, but not to any client application. Users can choose between three different predefined privacy levels (public, private, anonymous) which define the visibility of usage data, thus enhancing users' privacy. Moreover, users have the opportunity to delete any usage data at any time.

UTL is both learning design language and scenario dependent. Indeed, it was designed to represent tracks (i.e. context) specific to a predefined scenario, thus making difficult the analyses of context coming from heterogeneous contents and design languages. The query interface is not available at a

large scale, it is restricted to authenticated users (i.e. teachers).

The scope of **SBT-IM** is not to have an opened architecture to incorporate new context sensors: SBT-IM has a central architecture hardly coupled with the Moodle environment. Nevertheless, the raw context can be exported in OWL format to be shared across multiple systems, but the values of context are not kept up-to-date.

In **Learning Registry**, there are no restrictions about users or tools being able to publish or consume data, thanks to its Obtain, Harvest and Slice APIs. Nevertheless, these services offer only a specific list of paradata records, and does not provide any query functionalities (the upper layer has to implement this feature). Furthermore, the use of a distributed infrastructure to host and replicate metadata and paradata provides a high level of availability and scalability.

The architecture of **NSDL** is not very opened, since the paradata is included into the repository by harvest process: applications cannot use the framework to directly insert paradata. On the other hand, NSDL promotes the share and reuse of paradata through a search API and an OAI-PMH data provider. The first one offers a resource centric search of metadata and paradata, while the second provides metadata and paradata to subscribed applications. Finally, NSDL addresses users privacy through NSDL privacy policy [MK07].

As a conclusion of this survey, we notice that it is very difficult to design a system that entirely fulfills the whole set of criteria defined at the beginning of this chapter. On the one hand, Dyonipos and WildCAT have a highly flexible, expressive and extensible model, but they introduce scalability and sharing issues. On the other hand, KnowledgeTree handles the privacy issues but lacks flexibility, expressiveness or extensibility. Even if the CAM framework has an open architecture and highly supports context sharing, it lacks scalability and the ability to model context at a high abstraction level. SBT-IM provides a model to manage indicators, but deals with context specific to a single application. Finally, we can notice from Table 3.3 that most of the systems do not address privacy of users, even if they feel more and more uncomfortable with the sensitive data.

3.4 Conclusions

Chapter 2 has highlighted the need for AdWLEs to benefit from a maximum amount of observation data to perform relevant adaptation process. This chapter has described the context from our point of view: the user context includes the learner model, the environment context represents the learning materials and tools the user focused on, and the usage context express actions performed by users over contents and tools. In addition, a distinction has been made between the raw (resulting from direct interactions between users and systems/resources) and the inferred contexts (produced on the basis of the raw data).

We have also elaborated a set of criteria that are of most importance when context has to be shared and reused at a large scale, and established a survey of existing systems according to these requirements. In the second part of this document, we have exposed our proposals to meet these requirements.

Our context model should respect the design principles specific to context-aware applications: it does not need to be fully general; instead it should be characterized by a single backbone that applies to any application while providing semantic constraints applying to specific observed systems. [BCQ+07] stated that the practical applicability and usability of a model are often inversely proportional to the generality of the model: the more expressive and powerful it is, the less practical and usable. The raw context model should also feature extensibility. Indeed, the computational context of the user continuously evolves and comprises more and more applications. Therefore, the model should be able to federate heterogeneous contexts specific to existing applications, but also facilitate the integration of new contexts as they appear. In addition, the raw model should be enhanced by the possibility of defining and processing any inferred data, or indicator, on the basis of any element of the user, environment and usage contexts. Also, reuse of existing indicators should be optimized to make it easy for users to define new indicators on the basis of the existing ones.

The architectural design of our proposal to support the above model should adopt a distributed approach, since local and/or distant applications can be observed, and third party applications can remotely query the federated contextual data. This architecture should facilitate the indexation and retrieval of remote data by providing a dedicated set of tools, and components dedicated to the management of indicators should promote their reuse by adaptive applications. Furthermore, users should be provided with an easy-to-use mechanism to extend the context model according to their pedagogical needs. Since a large amount of context information may have to be processed, the architecture should be scalable as well: the number of data managed by the system should not influence the good behavior of the whole framework. The last important feature of our proposal, but not least, relates to confidentiality; since the context model includes the user profile and may depict personal information and/or sensitive data about users, it is of most importance to consider the privacy of the user according to the seven principles of privacy exposed in this chapter.

PART II - Share and Reuse of Context Metadata Resulting from Interactions between Users and Heterogeneous Web-based Learning Environments

Chapter 4 - Modeling the Web - based Learning Environments

Contents

Tracking is not new: in the area of applications, systems and networks management, knowing the state of a machine or equipment has been a major concern since 90's. Thus, a standard emerged in order to be able to supervise any network or computer system. The standard initiated by the Distributed Management Task Force (DMTF) brings a solution to unify management of distributed computing environments, and facilitates exchange of data across otherwise disparate technologies and platforms. The DMTF standard comprises a Common Information Model (CIM) to represent entities to be managed, and an associated Web-Based Enterprise Management (WBEM) architecture. The CIM is detailed next while the WBEM architecture is presented in the following chapter. Indeed, our framework to collect, share and reuse context data is based on DMTF's initiatives and experience.

4.1 The Common Information Model

The CIM is a conceptual view of the managed environment based on the Object paradigm. It is a hierarchical model that makes it straightforward to track and depict the complex interdependencies and relations among different managed objects. Such interdependencies may include those between logical network connections and the underlying physical devices, or those between an e-commerce transaction and the database servers processing this operation.

4.1.1 The Main Concepts of CIM

The CIM modeling approach specifies a meta-model on the basis of a common model of abstraction that describes a semantic knowledge of the managed world. The basic elements illustrated on Figure 4.31 are used to model entities to be managed:

- **Managed object / Class.** Object-oriented modeling is a formal way of representing something in the real world. The managed objects all inherit from the *CIM_ManagedElement* class and are specified by a set of properties and methods, while instances are representations of the objects in the real world;
- **Operation / Action.** An action is an operation that can be invoked on an element of the model and acting on the state/behavior of the matching real world entity;
- **Relation.** CIM suggests two types of relations:
  1. *Inheritance*: classes may inherit from one or several classes;
  2. *Association*: this relation binds at least two classes. Two specific types of associations have been defined: *CIM_Component* expresses a relationship of composition (i.e. the physical elements that are integrated within a computer), and *CIM_Dependency* reflects existential or functional semantics (i.e. a service running on a system is not available anymore if the system crashes).

4.1.2 The Levels of Abstraction

To meet needs of unification and extensibility, CIM offers a set of schemas divided into three distinct levels:

1. **The CIM Core model** provides a basic set of classes, associations and properties to describe any management area;

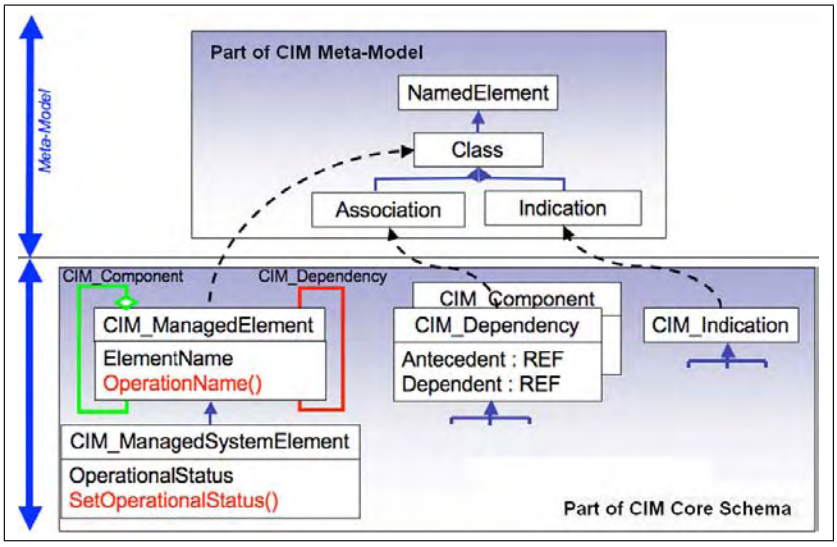


Figure 4.31: Basic elements of the CIM modeling approach.

2. **The CIM Common models** capture notions that are common to specific management areas, but independent of any particular technology or implementation. The actual schema comprises twelve common models appearing in Figure 4.32: Application, Database, Device, Event, Interoperability, Metrics, Network, Physical, Policy, Support, System and User;

3. **The CIM Extension models** represent technology-specific extensions of common models. Here, one can extend common schema in order to precisely describe a specific target area.

Figure 4.32: CIM core and common models.

Some extensions of the existing models are presented later in this chapter to specify a set of models dedicated to context management.

4.1.3 CIM Qualifiers

Qualifiers are values that provide additional semantics about classes, associations, methods, method parameters, properties or references. The main CIM qualifiers are the following ones:

- **Key** applies to a property and indicates that the property is (one of) the identifier of the class;
- **Description** provides a human-readable description of a method, property or class;
- **Values** represents an array of values that can be assigned to a property. This qualifier brings context constraints since it reduces the number of admissible values;
- **MappingStrings** applies to a property, and indicates the name of the matching entity as it appears in one or several other standards. It is very useful to map a format to another, since different views of the CIM models can be extracted according to any given standard or specific model.

CIM qualifiers are equivalent to Dublin Core qualifiers, in the sense that they refine the significance of certain elements. A refinement restricts the meaning of an element without changing it fundamentally.

4.1.4 The Motivations of our Choice

The CIM metamodel and its set of schemas represent an appropriate alternative to reach the set of criteria defined in the previous chapter and related to the modeling of context. Its object-oriented design offers multiple advantages:

- **Flexibility:** all classes modeling entities to be managed inherit from the abstract class *CIM_ManagedElement* described in the Core model. As the learning context differs from an application/resource to another, CIM offers an unified representation of heterogeneous contexts that are specific to various applications;
- **Extensibility:** the Extension models offer the opportunity of representing new contextual information whenever a specific need appears (e.g. a new learning application, resource or activity has to be observed);
- **Reusability:** the Common models define generic models specific to a given domain (Network, System, etc.) or area (space, finance, etc.). In the case of e-learning, a generic model based on existing generic classes such as *CIM_User, CIM_ApplicationSystem* or *CIM_Resource* would represent a basis to be reused and extended when one comes to design a specific context model;
- **Expressiveness and valid constraints:** thanks to composition relations and qualifiers, CIM brings expressiveness (i.e. resource can be composed of smaller chunks of data). On the other hand,

association relations allow to specify some constraints that have to be respected by the context entities;

- **An existing User model:** CIM includes a user profile through the User model which represents individuals together with their characteristics. This model can be further extended with specific properties, as shown in the next section;
- **An existing Metrics model:** as AdWLEs need to infer indicators on the basis of the collected data, management applications require inferred data called *metrics* to facilitate the management process. Therefore, CIM suggests a Metrics model to represent such data, thus providing an appropriate basis to specify e-learning indicators.

Since the CIM approach tackles all requirements related to the design of the context models, it obviously appear to be an appropriate candidate to achieve our goals. Therefore, on the basis of some existing CIM models, the next section presents our generic models specific to learning environments.

4.2 Representing Context of an AdWLE

According to our definition of context, our models make a distinction between the user, environment and usage contexts: the user context comprises the various characteristics of the users (interest, knowledge, goals, etc.), the usage context is composed of data about activities performed on resources and systems, and the environment context refers to data related to applications, systems, resources, etc. The resulting model appears on Figure 4.33: classes prefixed with "CIM" are part of the CIM Core and CIM Common models, while classes prefixed with "TEL" represent our own classes specific to technology enhanced learning [BVV+10]. The model and its associated classes are detailed in the next sections.

4.2.1 The User Context

The CIM User model suggests three main classes to describe users, that appear on Figure 4.34:

- **The *CIM_Person* class** is used to represent people, and holds their yellow and white pages. The yellow pages contain information such as the business category of the individual's organization, the employee number or the designated position of the individual. The white pages include traditional data such as the first name, last name, mail/postal address, etc.;
- **The *CIM_OtherPersonInformation* class** provides more detailed information about a person (the name of the organization the person is enroled in, password, preferred language, etc.). It is linked to *CIM_Person* through the *CIM_MorePersonInfo* association;

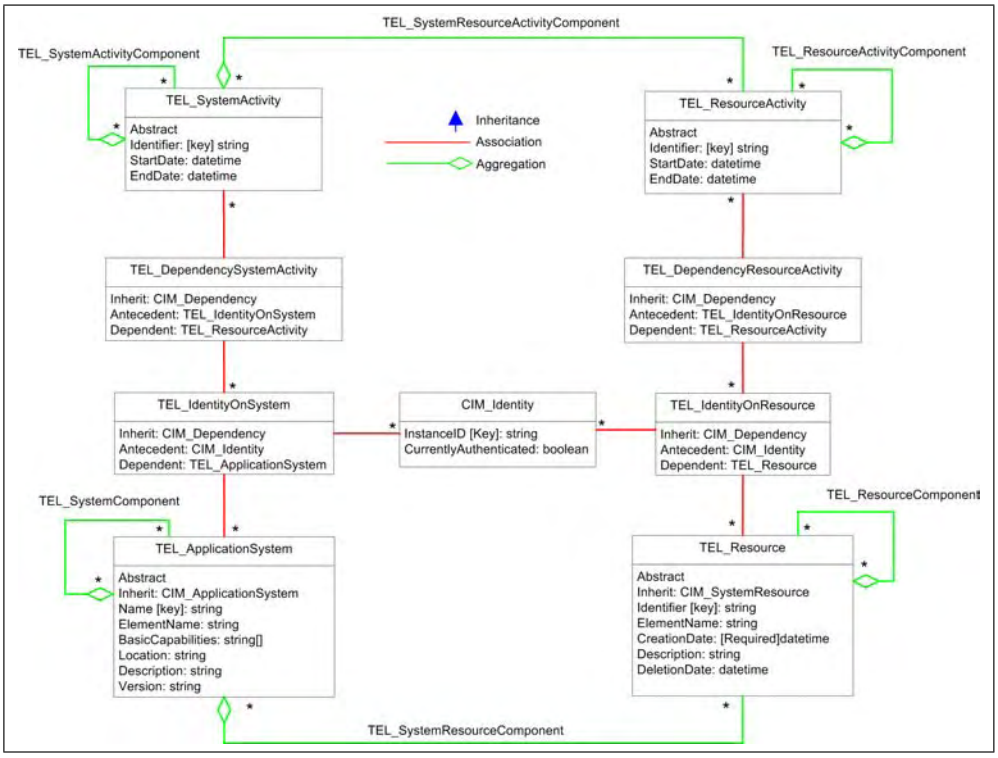


Figure 4.33: The context model.

- **The *CIM_Identity* class** is the entry point into the User model, as it acts as a reference to the other classes. This class is linked to *CIM_Person* through the *CIM_AssignedIdentity* association.

The CIM User model contains general information about a user with a focus on the organizational aspects, but doesn't address the pedagogical or learning point of views. As a consequence, the CIM User model was extended by [RVB10] to consider the missing information. The main class of the extended model is the abstraction *TEL_ProfileCore* that represents the top-level class to design any profile specific to TEL actors (e.g. learners, teachers). This class ensures extensibility and openness, and covers any profile that may be required to optimize any TEL application or system. Figure 4.34 details the core learner profile represented by the class *TEL_LearnerCore*. For interoperability reasons, this model ensures a full compatibility with the Learner Information Package (LIP) specification [LIP08] and separates a learner profile into four subprofiles. The *Identification* profile relates on demographic information of the learner, and integrates attributes specified within the *Identification* category of the LIP specification. The *Cognitive* profile measures learner performances, goals and competencies; this sub-profile specifies most of the LIP categories as an enumeration of associative arrays. The *Metacognitive* profile aims at measuring how a learner thinks about his/her cognitive skills: learners who cannot monitor accurately cannot correct errors and as a consequence, they process information less efficiently than self-monitored learners. The *Preference* profile includes three LIP categories and details

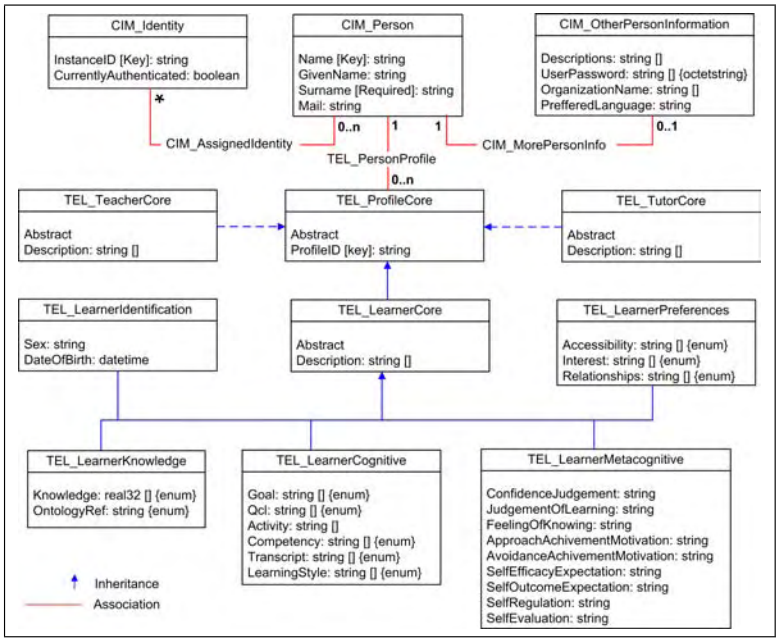


Figure 4.34: The user context.

information about what a learner would prefer to be applied during a distant learning session, or about his/her general interests.

The model detailed in [RVB10] didn't describe the knowledge levels of a student regarding concepts of a given ontology. Since Chapter 2 showed that this information can be crucial for some categories of AdWLE, we introduced the *TEL_LearnerKnowledge* class. The *Knowledge* array stores a value for each concept of the ontology referenced by *OntologyRef* and reflects how well the student masters a concept. A concept is not unique, the same concept may exist in many domain ontologies. The value of the concept is different for each ontology, since the relationships between concepts may be different too: in one ontology, a concept may depend on 2 concepts, while in another ontology the same concept depends on only one concept. The relationships between concepts are represented. The order of the concepts in the knowledge array is given by one of the domain graph browse methods (i.e. Depth-first or Breadth-first). Nevertheless, only one type of relationship can be represented at a time (e.g. requires, sub-concept of). The coherence of one concept among different domains may be managed through domain ontologies merging techniques [CBY11].

The resulting model comprises all entities of the student model depicted in Figure 2.4. The domain-dependent component of the student model, *knowledge*, is represented by the *TEL_LearnerKnowledge* class. Thus, multiple instances of this class represent the knowledge of the student for different learning areas. Furthermore, the goals, background and learning style of the student are represented within the class *TEL_LearnerCognitive*. Finally, interests of the student are described through the class *TEL_LearnerPreference*, while demographic information is mostly provided by several native CIM classes of the

user model. Let us note that this user model is not restricted to learners, thanks to the core profile that may be extended to address specificities of teachers, tutors or even researchers.

4.2.2 The Environment Context

As already mentioned, the environment context refers to data describing the tools and resources the user interacts with. The environment model is presented in Figure 4.35, and is linked to the user context through the class *CIM_Identity*.

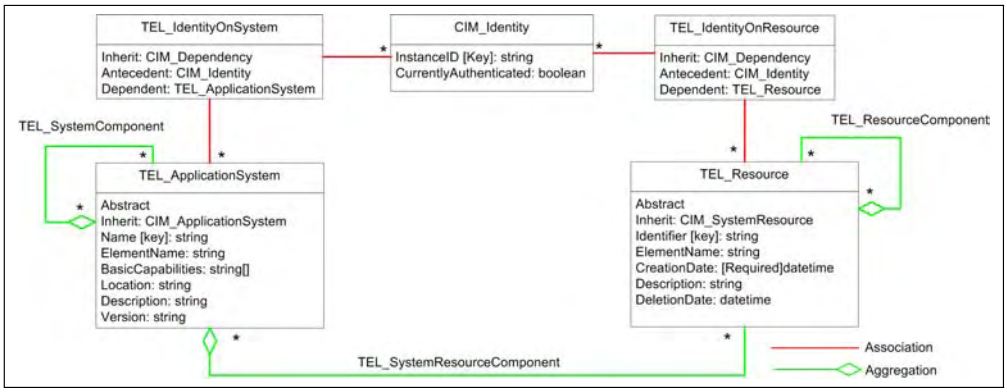


Figure 4.35: The environment context.

4.2.2.1 TEL Application Systems

The native *CIM_ApplicationSystem* class represents an application or software system characterized by a particular business function, and that can be managed as an independent unit composed of various elements. In order to represent applications specific to the learning area, we extended this class through the class *TEL_ApplicationSystem* depicted on Figure 4.35. The main elements describing a TEL application are:

- **Name** is the machine-readable key of the class, and is used to uniquely identify an application or system within the learning environment;
- **ElementName** specifies a user-friendly name for the object. If the *Name* property has a non understandable value, *ElementName* may be used for humans to identify a certain application;
- **BasicCapabilities** denotes the role of the application. An appropriate value of this attribute may be “store learning objects” for a learning object repository, “deploy learning resources” for a learning management system, or “visualize web pages” for a web browser;
- **Location** locates the system on Internet. This property is not required, thus allowing to represent desktop applications;

- **Description** provides a textual description of the object;
- **Version** represents the version of the application.

This generic class will serve as a basis for designing specific applications used to study in a given learning environment, as shown in Chapter 7.

4.2.2.2 The Learning Resources

The class *TEL_Resource* describes resources integrated within learning environments. It inherits from the native class *CIM_SystemResource* which models any entity managed by an operating system or software application. The main properties of this class are the following ones:

- **Identifier** represents the key of the class and helps to uniquely identify a learning resource;
- **ElementName** depicts a user-friendly name. In our area, this property may be the title of the learning resource;
- **CreationDate** refers to the date when the resource was created, if available;
- **Description** provides a textual description of the resource, and acts as the LOM. Description metadata;
- **DeletionDate** points to the date when the resource was deleted, if a user removed a resource from a system. If this property is not empty, the real resource doesn't exist anymore but the matching instance represents its history;

Let us mention that the *TEL_Resource* class does not integrate the learning resource metadata (e.g. LOM, DC, etc.). Instead, this metadata is accessible through the learning system hosting the resource, and more precisely through the *Location* property. Properties such as language, goal, age, difficulty and audience can be retrieved from the repository where the resource is hosted.

Any resource included into a learning application will inherit from this root class, and extended to depict its specificities.

4.2.2.3 Associations and Compositions

The association classes express the existing relationships between the various components of a learning environment. Our environment context comprises two dependency relationships and three composition associations:

- **TEL_IdentityOnSystem** associates a user with the learning system(s) he/she interacts with. The *-* cardinality signifies that a user can interact with multiple systems, and that a learning system can deal with multiple users. Using this association, it is possible to know which are the learning systems a specific user interacted with, or the users who interacted with a given learning system;
- **TEL_IdentityOnResource** is similar to the previous one, but associates a user with the learning resources. Using this association, it is possible to know which are the learning resources a specific user interacted with, together with the users who interacted with a specific learning resource;
- **TEL_SystemComponent** is used to represent applications at different levels of granularity, since one application can be composed of smaller sub-applications;
- **TEL_ResourceComponent** is used to represent resources at different levels of granularity, since a given pedagogical resource can be composed of smaller pedagogical sub-resources (several paragraphs compose a document, several questions compose a quiz, etc.). In order to add detailed semantics between resources at a higher level of details, this class may be extended;
- **TEL_SystemResourceComponent** expresses the attachment of a resource to one or several learning systems. It has been modeled as an abstract class to avoid ambiguity between systems and resources. Thus, for more detailed semantics, this class has to be extended (see Chapter 7).

The use of relations brings a series of advantages. First, with dependencies we know at any moment which are the learning resources and systems a learner is interacting with, while using compositions it is possible to browse learning resources and systems at different levels of granularity. Another advantage of the composition relation is that it ensures some context constraints. As an example, by specializing the class *TEL_SystemResourceComponent*, it is possible to impose some constraints that context entities have to conform with: a learning object *is stored into* a learning object repository, while a courseware may *be deployed by* a learning management system. Furthermore, model designers are in charge of appreciating the level of details they want to achieve.

4.2.3 The Usage Context

The usage model comprises the activities (or actions) that users can perform over learning systems and resources. We distinguish two generic activity classes, one dedicated to activities on learning resources and the other one to activities on learning systems.

4.2.3.1 Resources Activity Model

The main class of this model is *TEL_ResourceActivity* (see Figure 4.36). This abstract class represents the highest level of abstraction of activities over learning resources. The next level of abstraction is provided

by classes describing activities specific to a given type of learning resource, followed by classes defining concrete activities from the real world (i.e. consultation, download, rating, etc.). The basic properties of the class *TEL_ResourceActivity* are the following ones:

- **Identifier** is the key of the class and uniquely identifies an instance of activity;
- **StartDate** specifies the date when the activity started. Correlated with the *EndDate* property, it is possible to calculate the duration of an activity (i.e. the time spent to solve an exercise, to read a document, etc.);
- **EndDate** depicts the date when the activity ended. If *StartDate* and *EndDate* have the same value, then the matching activity has no duration (i.e. click on a link, log into an application, etc.).

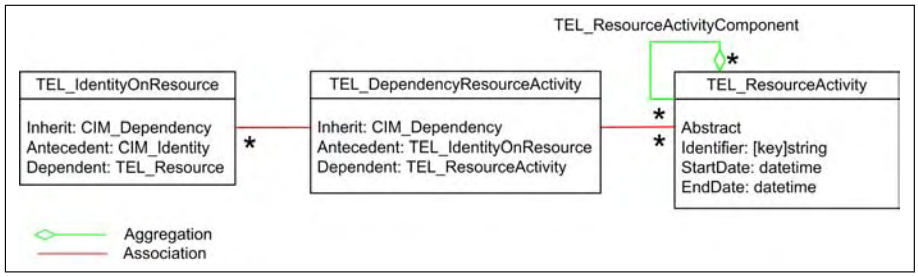


Figure 4.36: Resource activity model.

The connection to the environment context is provided by the association class *TEL_DependencyResourceActivity*. This class associates an instance of *TEL_IdentityOnResource* (which links a user and a resource) with an instance of *TEL_ResourceActivity*. In this way, a tuple user-resource-activity is represented in a unique way. By exploiting this association, various information is made available: the whole set of activities performed by a given learner on a specific learning resource, or the set of resources on which a given learner performed a specific activity, or the learners who performed a specific activity on a given learning resource.

As an activity can be composed of sub-activities, the abstract class *TEL_ResourceActivityComponent* allows to define composition relationships between different types of activities. It has a *-* cardinality, since an activity can be composed of multiple activities, and an activity can be part of various activities of higher level(s).

4.2.3.2 Systems Activity Model

This model (see Figure 4.37) is symmetric to the resource activity model. *TEL_SystemActivity* is the main class and represents the highest level of abstraction to describe activities performed on applications. Such activities may include the login or logout of the learner on a learning platform, or the configuration of a learning system. The model is extensible so any kind of activities may be described. Just

like the resource activity model, the main attributes of *TEL_SystemActivity* are *Identifier*, *StartDate* and *EndDate*, with the same meaning. In the case of learning systems, *StartDate* and *EndDate* may indicate the duration of a learning session, or how long a learning system was operational.

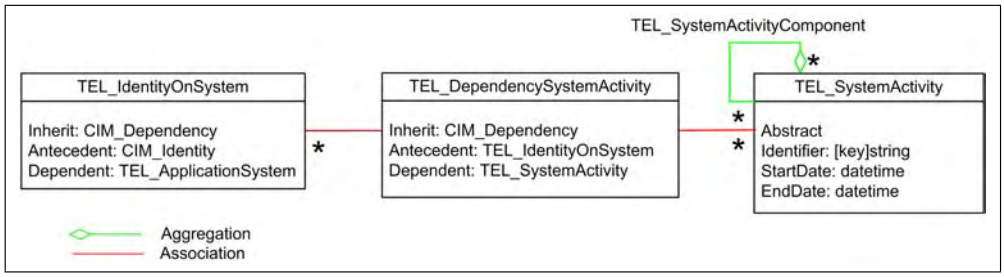


Figure 4.37: System activity model.

The connection to the environment context is provided by the association class *TEL_DependencySystemActivity*. This class associates an instance of *TEL_IdentityOnSystem* (which links a user with a learning system) with an instance of *TEL_SystemActivity*. In this way, a tuple user-system-activity is represented in a unique way. By exploiting this association various information is made available: the whole set of activities performed by a given learner on a specific learning system, or the set of systems on which a given learner performed a specific activity, or the learners who performed a specific activity on a given learning system.

The abstract class *TEL_SystemActivityComponent* expresses the same semantics than *TEL_ResourceActivityComponent*, but relies on system activities. Finally, the relation *TEL_SystemResourceActivityComponent* (illustrated on top of Figure 4.33) translates the fact that an activity on a resource may be part of a higher level activity operated on a system.

4.2.3.3 Input Devices

The range of web-capable devices is nowadays extremely wide and in continuous increase: users are not using anymore only a computer to study, instead they access learning content through a wide variety of mobile devices (e.g. PDA, smartphones, notebooks, etc.). Thus, it is of interest for an AdWLE to know what kind of device(s) a given user has access to, in order to adapt not only to the knowledge, interests and preferences of the user, but also to his/her devices.

The CIM model suggests the class *CIM_UserDevice* to represent users' devices. These are devices that allow users to input, view or hear data through a computer system. Natively, two user device subclasses are defined: *CIM_Keyboard* specifies the characteristics of the keyboard and *CIM_PointingDevice* represents the pointing device, both used for data input.

In our area, the *CIM_UserDevice* class and its sub-classes represent the devices used by users to perform an activity over a learning resource or system. The main properties of *CIM_UserDevice* and its

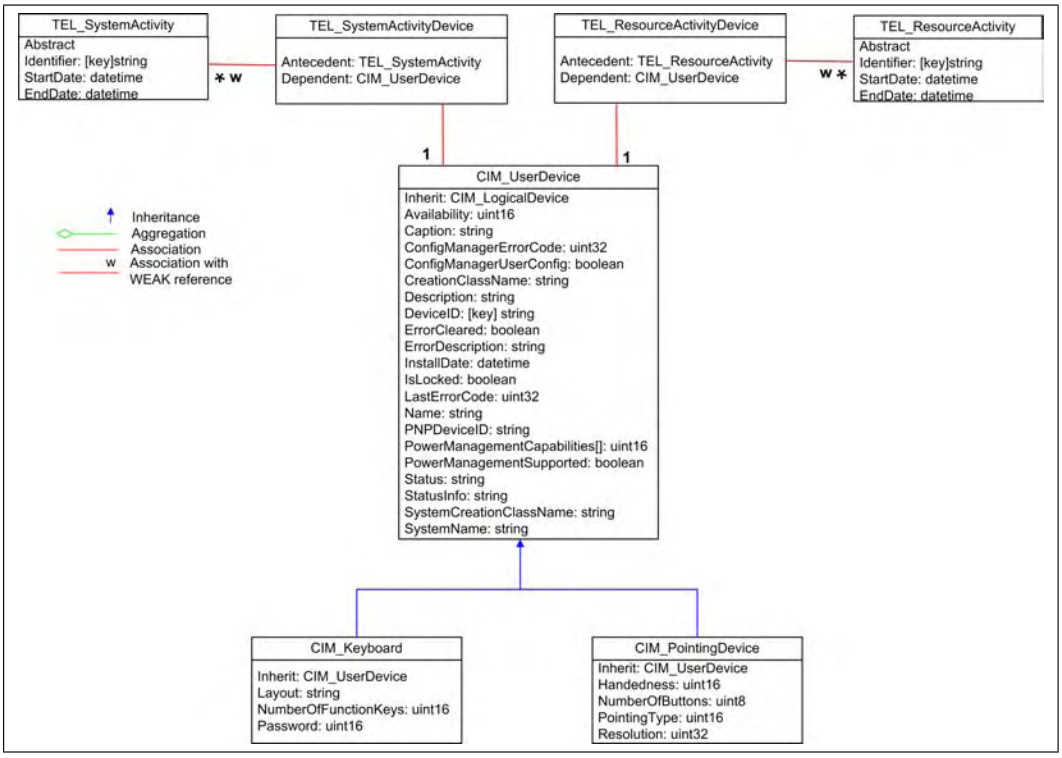


Figure 4.38: Input device model.

two sub-classes illustrated in Figure 4.38 are the following ones:

- **DeviceID** is the key of the class and uniquely identifies a specific device;
- **InstallDate** reflects the date when the device was installed on the user host;
- **Availability** depicts the status of the device at the moment the activity was performed. CIM defines seventeen possible values for this property (including running/full power, off line, off duty, not installed, etc.);
- **IsLocked** prevents any user input or output with the matching device (if set to TRUE);
- **SystemName** references the name of the system hosting the device;
- **Layout** indicates the format and layout of the keyboard (examples of keyboard layout can be QWERTY or AZERTY);
- **Password** indicates whether a hardware-level password is enabled on the keyboard to prevent local input;
- **Handedness** specifies the configuration of the pointing device for right-hand or left-hand operation;

- **PointingType** represents the type of the pointing device. Possible values are other, unknown, mouse, track ball, track point, glide point, touch pad, touch screen or mouse optical sensor.

As shown in Figure 4.38, a device is associated to a resource activity through the class *TEL_Resource-ActivityDevice* (and to a system activity through the class *TEL_SystemActivityDevice*). Both classes have w* - 1 cardinality, since one particular activity instance cannot be performed by multiple devices; at the opposite, a given device can be used to perform multiple activities. Using these two associations, the list of devices a given user used to perform activities can be established and appropriate learning content may be further provided depending on the current device(s) capabilities. Considering the users' devices into our usage context is an important feature, since AdWLEs are able to perform adaptation according to specific devices available to users.

4.3 The Indicator Model

The models presented until now are specific to raw contextual data, and thus not well-adapted for inspection and interpretation by teachers. To have a meaning for them, this raw context has to be processed to infer more pedagogical-oriented measures. This section is dedicated to our proposition related to a generic TEL indicator model.

4.3.1 The CIM Metrics Model

The native CIM Metrics model is dedicated to the specification and retrieval of metric information. A metric, in the system and network management area, is used to explicitly express understandings about resources that are monitored. Metrics help administrators to faster and easier react and set up rescue plans. Examples of metrics characterizing systems and networks are the total number of instructions processed by a computer system or the load of a router during a specific period of time. The CIM Metrics model comprises two classes illustrated on Figure 4.39:

- **CIM_BaseMetricDefinition** behaves as a pattern that specifies the semantics and usage of a metric (e.g. its metadata). *CIM_BaseMetricDefinition* does not capture the value of the metric, instead the class *CIM_BaseMetricValue* holds this data. The purpose of *CIM_BaseMetricDefinition* is to provide a convenient mechanism for introducing a new metric definition at runtime and capturing its values in a separate class. Additional metadata for a metric can be provided by sub-classing *CIM_BaseMetricDefinition*, but the most important properties of this class are:
  - **Name** gives a descriptive name of the metric (e.g. request rate);
  - **DataType** represents the data type of the metric (e.g. boolean, datetime, numeric);

  - **ProgramaticUnits** reflects the measurement unit of the metric (e.g. bites per second);
  - **GatheringType** indicates when the metric values must be calculated by the underlying instrumentation. The possible values are: 1- the metric is calculated just once, at the moment of definition, 2- whenever a resource attached to a metric is updated, 3- periodically, or 4- on request;
  - **SampleInterval** indicates the time between two calculations (it applies only when metric values are periodically calculated).

- **CIM_BaseMetricValue** acts as a container of values associated with the metric definitions. One metric value is contained in each instance of this class, and each instance is associated with a *CIM_BaseMetricDefinition*. The main properties of this class are:
  - **TimeStamp** indicates the time when the value of a metric has been computed by the underlying instrumentation;
  - **Metric Value** is the value itself, stored as a string;
  - **Volatile** indicates if a new instance must be created when a new measurement occurs, or if the existing instance must be updated.

In addition, several associations are defined. They relate any type of managed resources (as stated before, all classes inherit from the class *CIM_ManagedElement*; therefore, all classes of the context model implicitly inherit from it as well), metric definitions, and metric values:

- **CIM_MetricDefForME** defines which *CIM_BaseMetricDefinition* objects apply to a given entity (i.e. *CIM_ManagedElement*). A single definition may apply to one or several managed entities, and a given element can be characterized by an unlimited number of metric definitions;
- **CIM_MetricInstance** links a metric value to a metric definition. A value applies to a single definition, whereas a definition may be linked to several values;
- **CIM_MetricForME** allows finding all *CIM_BaseMetricValue* objects available for a given *CIM_ ManagedElement*. As the *CIM_MetricDefForMe*, a single value may apply to one or several managed elements, and a given entity can be characterized by an unlimited number of metric values.

On the basis of the CIM Metrics model, we designed a generic model dedicated to indicators specific to technology-enhanced learning [BVB12].

4.3.2 The Indicator Model

As stated above, a metric applies to any *CIM_ManagedElement*. Since all classes of our context model inherit from this class, a metric can be defined for any element of the user, environment and usage contexts, no matter if it is a class, dependency or composition.

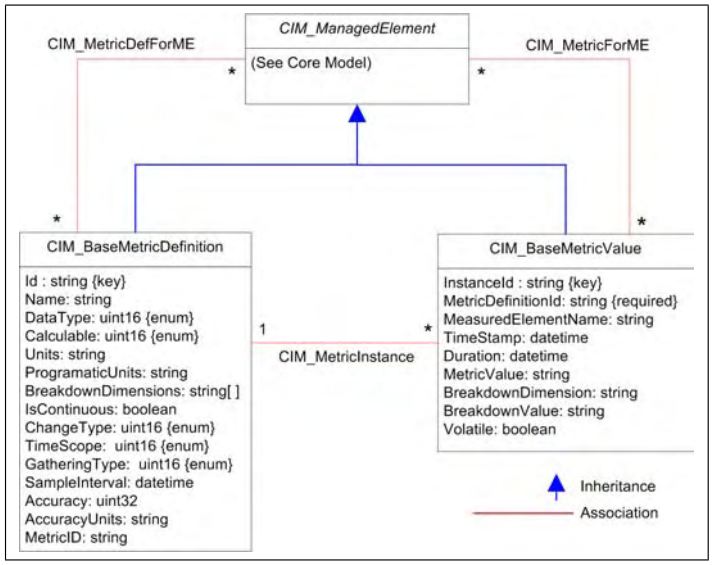


Figure 4.39: CIM Metrics model main classes.

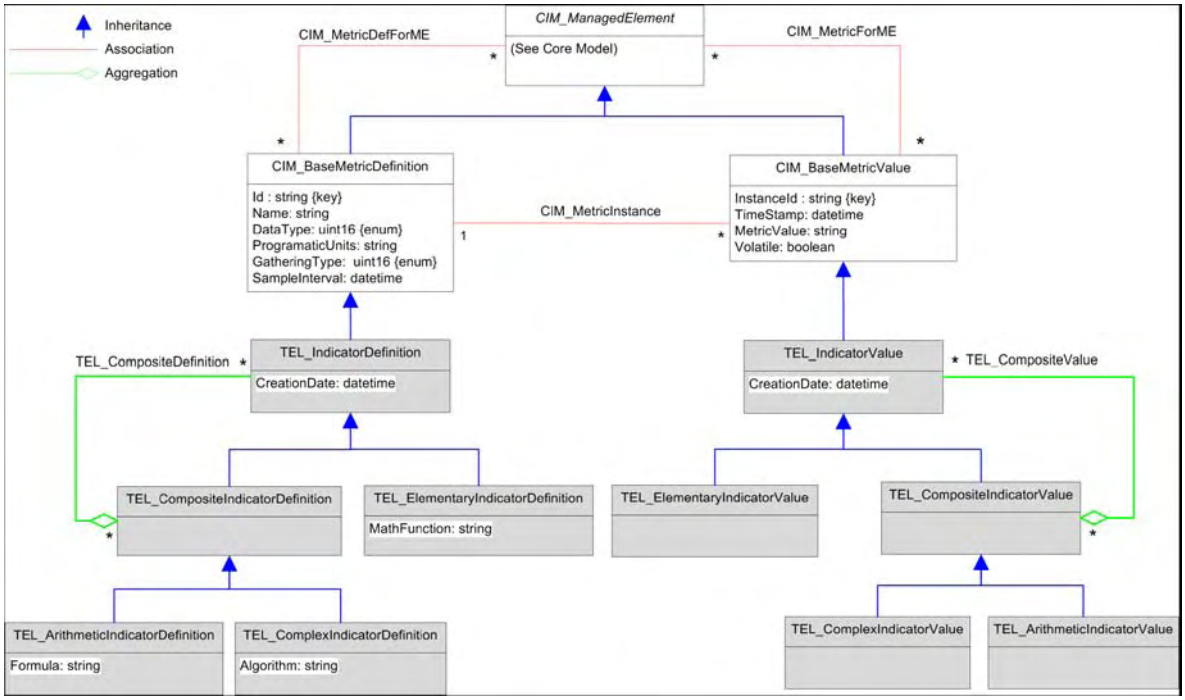


Figure 4.40: Indicator model.

Our model dedicated to TEL indicators is illustrated on Figure 4.40, where the two top level classes (*TEL_IndicatorDefinition* and *TEL_IndicatorValue*) directly inherit from the matching native CIM classes. Under the definition class, we made a distinction between an elementary indicator and a composite indicator in order to decompose complex indicators into simpler ones that may be reused to build other complex indicators.

4.3.2.1 The Elementary Indicators

An elementary indicator (class *TEL_ElementaryIndicatorDefinition*) is an indicator calculated straight from the raw data stored into our models. The value of such an indicator (class *TEL_ElementaryIndicatorValue*) should be expressed as a string, and results from simple mathematical functions (min, max, avg, sum, etc.) reflected by the *MathFunction* attribute. Examples of elementary indicators include the total number of activities performed over a learning resource, the average time spent by students to solve a problem, or the highest note taken by a student in one semester.

4.3.2.2 The Composite Indicators

A composite indicator is calculated on the basis of elementary or other composite indicators, and is defined using the class *TEL_CompositeIndicatorDefinition* or one of its sub-classes. To date we defined arithmetic and complex indicators, but other high level composite indicators can be defined as well.

4.3.2.2.1 The Arithmetic Indicators

Such an indicator is the result of complex mathematical operations applied to two or more indicators (i.e. elementary and/or composite identified through the association *TEL_CompositeDefinition*). A formula expresses how the indicator value must be calculated, and stands on some basic mathematical operators.

The calculation of an arithmetic indicator thus assumes an *a priori* definition and calculation of the elementary or other complex indicators involved in the calculation process. Once an arithmetic indicator is defined, the most recent values of the underlying indicators are taken into account, and the result of the computation is stored as an instance of the class *TEL_ArithmeticIndicatorValue*. An example of arithmetic indicator is the sum between the number of consultations of a learning resource, and the number of downloads. Two elementary indicators must be calculated to respectively calculate the number of consultations and downloads before the arithmetic indicator can be defined on their basis.

The arithmetic indicators apply when the number of underlying indicators is low or *a priori* defined, but are not appropriate when this number is high or *a priori* unknown. Thus, to allow the definition and calculation of such indicators, and to enhance the flexibility of our model, we introduced the complex indicators.

4.3.2.2.2 The Complex Indicators

This type of indicator (class *TEL_ComplexIndicatorDefinition*) cannot be calculated through the elementary nor the arithmetic indicators. A string property indicates the algorithm leading to the indicator value, thus offering the total freedom to elaborate advanced indicators which rely on a larger or unknown number of intermediary indicators. The *Algorithm* parameter may interact with the whole set of models described in this chapter, and thus use any context data to calculate as many intermediary indicators as needed.

4.4 Conclusions

The models presented in this chapter have described context at two levels of details: raw level resulting from interactions between users and applications, and inferred level built on the basis of the raw level.

The raw model is highly expressive thanks to various associations and composite relations between the user, environment and usage contexts. Our model is not application-bounded, since multiple tools and applications can be represented; it is not fully general either, thanks to various constraints such as a fixed structure of the root models (classes presented in this chapter cannot be modified) and some predefined data types that prevent the modeling of any entity. The raw model tries to reach the best generality-usability compromise; it offers a unified view of contexts describing heterogeneous applications and resources. Moreover, the extensible character of CIM provides our model with the capacity of integrating new contexts on the fly when new needs appear, as demonstrated in Chapter 7.

The model dedicated to the design and calculation of indicators, based on any element of the raw context, proposes, in addition to statistical and arithmetical indicators, the possibility of defining indicators characterized by a high level of complexity. This decomposition makes it easy to reuse intermediate indicators to build similar indicators, whereas the aggregation relations we introduced add semantics about their level of granularity. Even if the design of complex indicators may be difficult, once they are defined, they can be easily reused to apply on other entities of the raw context, thanks to the clear distinction of their definition and value(s). On the other hand, this distinction facilitates their reuse by adaptive learning environments: the metadata describing the definition of an indicator makes it easy to precisely identify the nature and objective of the inferred data. Finally, several values can be assigned to the same definition and/or raw element, thus offering the opportunity of retrieving the history of a given indicator for a given context element; the four different ways of keeping values up-to-date that can be implemented by the underlying instrumentation (according to a dedicated attribute) consolidate this process.

In order to support the context models and offer facilities to exploit the data, an adequate architecture providing a set of appropriate tools has to be defined; the next chapter presents our proposition.

Chapter 5 - The Global Architecture

Contents

In this chapter we present a four-layered architecture inspired from the Web-Based Enterprise Management (WBEM) architecture designed by the DMTF, and which supports the context models while fulfilling the architectural criteria and privacy principles defined in Chapter 3.

5.1 The WBEM Management Architecture

Figure 5.41 illustrates the management gap tackled by WBEM: on the left hand side are operators (something or someone giving commands to the management system), and on the right hand side is a device containing hardware, software and services. A mechanism is needed to connect these entities to allow operators to configure and request the heterogeneous elements composing the device, and to receive alarms and events occurring on it. The state of the cooling fan within the device might actually be accessed by reading the top two bits of a given register. This is something that the operator doesn't want to know, he/she simply wants to get the state of the fan. The aim of the WBEM architecture is to fill this management gap by hiding the details from the operator.

In Figure 5.42, a WBEM server has been introduced. The WBEM server acts as a broker between various independent elements:

- The **Client and listener** represent the operators. They are able to send commands and receive responses, events and alarms that occurred on the managed resource. Clients and listeners may

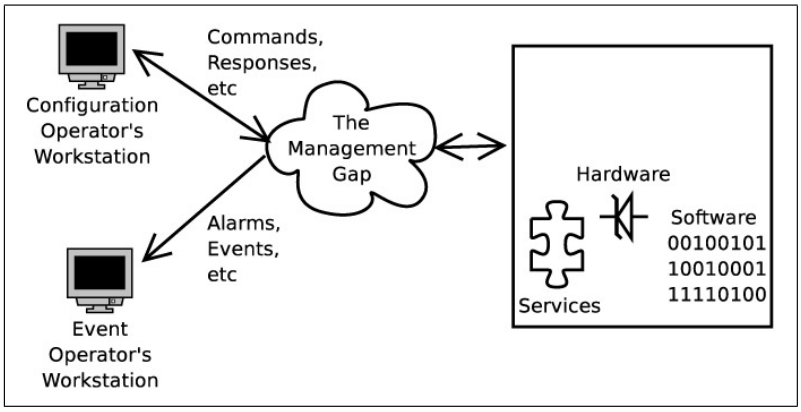


Figure 5.41: The management gap [Hob04].

reside on the resource being managed, on the operator's workstation, or anywhere else;

- **Providers** interact directly with the hardware, software and services of the managed resources, and translate an abstract request ("return the state of the fan") into a specific command ("read the top two bits of a specific register");

- **A repository** contains the management knowledge (information provided by the providers about the hardware, software and services of the managed resources) represented as CIM *instances*. The CIM *classes* are introduced into the repository through text files compliant with the Managed Object Format (MOF); Appendix A provides an example of the MOF file representing the class *TEL_LearningObject*.

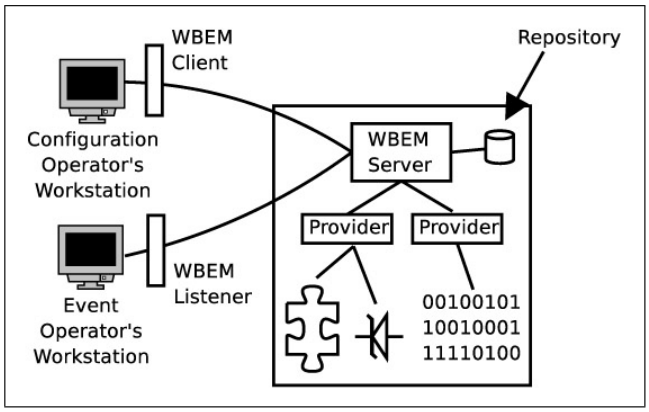


Figure 5.42: WBEM server components [Hob04].

The heart of the WBEM server is the CIM Object Manager (CIMOM). It orchestrates the workflow between the above entities by redirecting commands and responses or alarms, and ensures isolation of both WBEM clients and providers that can be designed independently of each other to enhance the management system.

Starting from the WBEM architecture, we designed our own architecture to collect and share contextual data produced by heterogeneous AdWLEs:

- A repository stores the context models presented in Chapter 4 together with their instances;
- The various AdWLEs represent the elements to be managed, where specific sensors act as WBEM providers;
- The WBEM clients and listeners are adaptive components of the AdWLEs querying the repository for adaptation purposes.

5.2 The Global Architecture

In order to design our architecture, we considered both the six-layered architecture presented in Chapter 3 and the WBEM architecture. This choice was influenced by two factors. On one hand, our framework is meant to collect data from heterogeneous AdWLEs which are widely spread on various locations. On the other hand, it aims at sharing the collected context with distant applications located worldwide. The resulting framework adopts a distributed approach and comprises four levels depicted on Figure 5.43 that make it possible to collect and share context metadata from/with heterogeneous AdWLEs located worldwide:

- The **context layer** is responsible for storing the context metadata and comprise additional components featuring indicator management facilities;
- The **middleware layer** offers an easy access to the context layer through a set of services;
- The **AdWLE layer** represents the learning environments from which context metadata are collected and reused for adaptation purposes;
- The **user layer** includes e-learning actors interacting with the learning systems together with their device(s).

Each of these layers is detailed in the next sections below.

5.2.1 The Context Layer

The context layer illustrated in Figure 5.44 is conform to the core components of the WBEM proposal: a repository acts as a database to store the context data, and a manager exposes several interfaces and APIs to easily exchange information with the repository; moreover, additional components are dedicated to indicators.

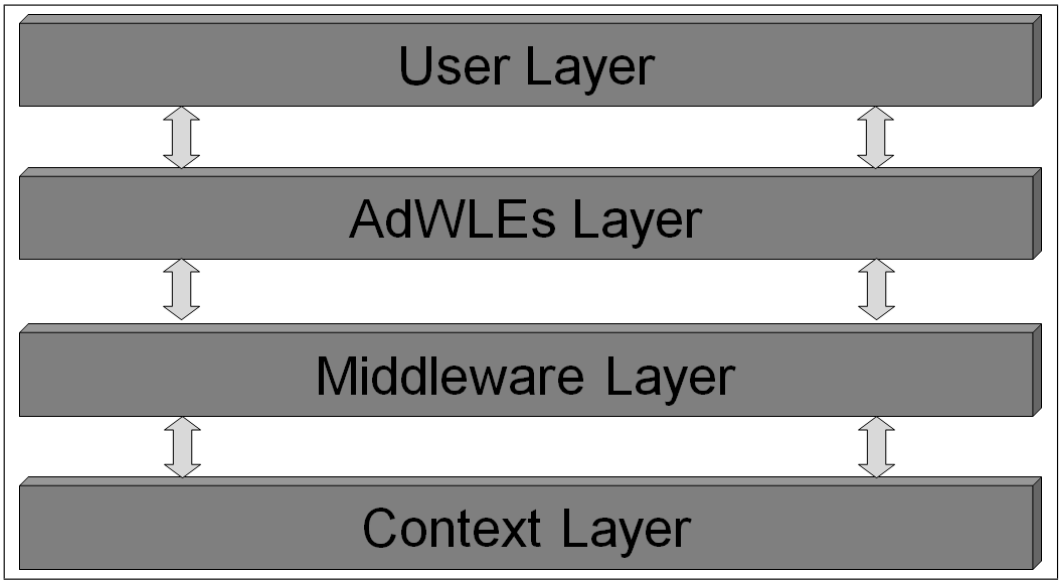


Figure 5.43: The global architecture.

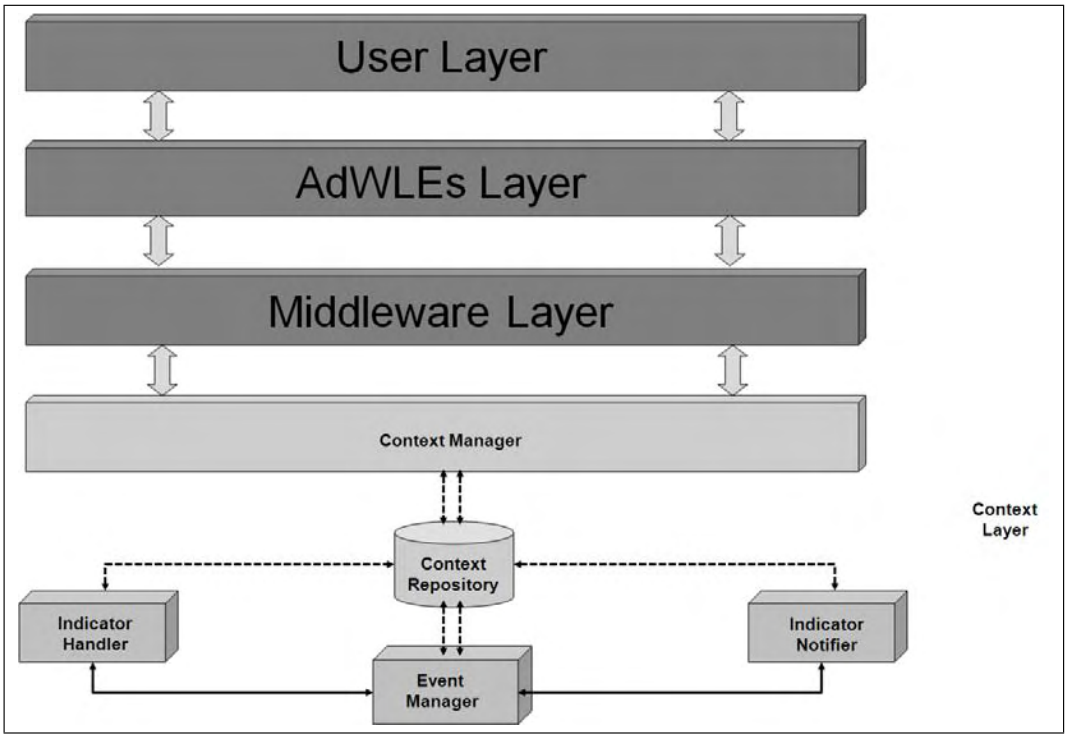


Figure 5.44: Details of the context layer.

5.2.1.1 The Core Components

As illustrated on Figure 5.45, the **context repository** contains both the context models presented in the previous chapter (as classes) and the instances associated to these definitions. The **context manager** interacts with the repository, and is responsible for the creation, modification and deletion of the data hold by the repository by sending queries expressed in a specific language. Thus, one of the main feature of the manager is to keep this data consistent.

To handle indicators, our approach is based on the publish/subscribe paradigm. The aim of this architecture is to push information to applications interested in a specific topic or event. In our framework, indicators definitions and values are delivered, at the right time, to any adaptive component interested in the analysis of users' behavior. Thus, in addition to its core components, the context environment comprises a set of entities dedicated to indicator management.

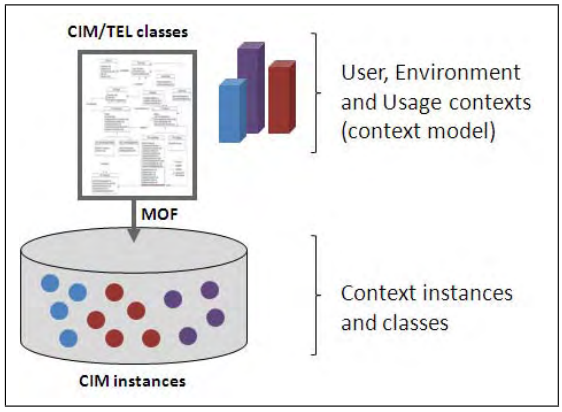


Figure 5.45: The Tracking Repository.

5.2.1.2 Components Required to Manage Indicators

The **event manager** has a double role. First, when a new *CIM_MetricDefForME* association is created between an instance of *TEL_IndicatorDefinition* and an instance of one of the classes of our models (for which the indicator is calculated), it notifies the Indicator Handler responsible for its calculation (see below). On the other hand, when the value of an indicator is created or updated (i.e. calculated), it sends a notification to the Indicator Notifier so that external actions can be executed.

The **indication handler** is responsible for calculating values of indicators according to their definition, and ensures the creation or modification of the matching instances (*TEL_IndicatorValue*, *CIM_MetricForME* and *CIM_MetricInstance*) into the context repository. Depending on the type of indicator, this component performs different process: in case of elementary indicators, the simple mathematical function is executed to calculate elementary values, whereas in case of complex indicators, the

values of the intermediary indicators composing the formula are processed and then the operators or advanced algorithms are appropriately applied.

The aim of the **indicator notifier** is to allow the execution of actions outside the repository whenever it receives a notification from the event manager: it invokes the indicator service of the middleware layer each time a new indicator value is calculated by the Indicator Handler. At its turn, the Indicator Service will notify the interested applications.

5.2.2 The Middleware Layer

The aim of the middleware layer is to bridge the gap between the tools of the adaptive learning environment and the context layer by offering an easy access to the repository. This intermediate layer is designed as a Service Oriented Architecture (SOA), since this paradigm brings several advantages [RLSB08]:

- **Interoperability:** this is the most important benefit of SOA. Services are platform-independent thanks to the use of standard-based protocols, but also programming languages-independent, making it possible for heterogeneous applications (in terms of programming language) to communicate and exchange information;
- **Aggregation/reusability:** SOA divides the complexity of services in sub-services with smaller complexity which handle simpler functionalities. Complex services can then be built on top of simpler ones, and a service which provides a feature (e.g. query service) can be reused to build other services;
- **Deployability:** services are deployed over distributed standard technologies, making it possible to access services running on the other side of the globe. Also, thanks to the use of proven community standards such as WS-Security [BMPS09], various mechanisms can be set up to ensure security of information during its transportation.

Using SOA, web-based tools are able to easily provide and/or retrieve context stored into the repository, thanks to the following services of the middleware layer appearing on Figure 5.46:

1. The **session management service** allows consumers to establish a session required before any communication can take place with the provider of one of the other services. According to the type of session, various qualities of services can be provided;
2. The **search service** receives queries from consumers and communicates with the context manager to retrieve the matching information into the context repository;

3. The **insert service** allows the indexation of new contextual metadata into the repository (through the context manager);

4. The **model management service** offers a set of methods to facilitate extension and modification of the context models;

5. The **indicator service** offers a means for web-based applications to subscribe (or unsubscribe) to indicators of interest, and then notifies subscribers when the value of an indicator of interest is (re)calculated.

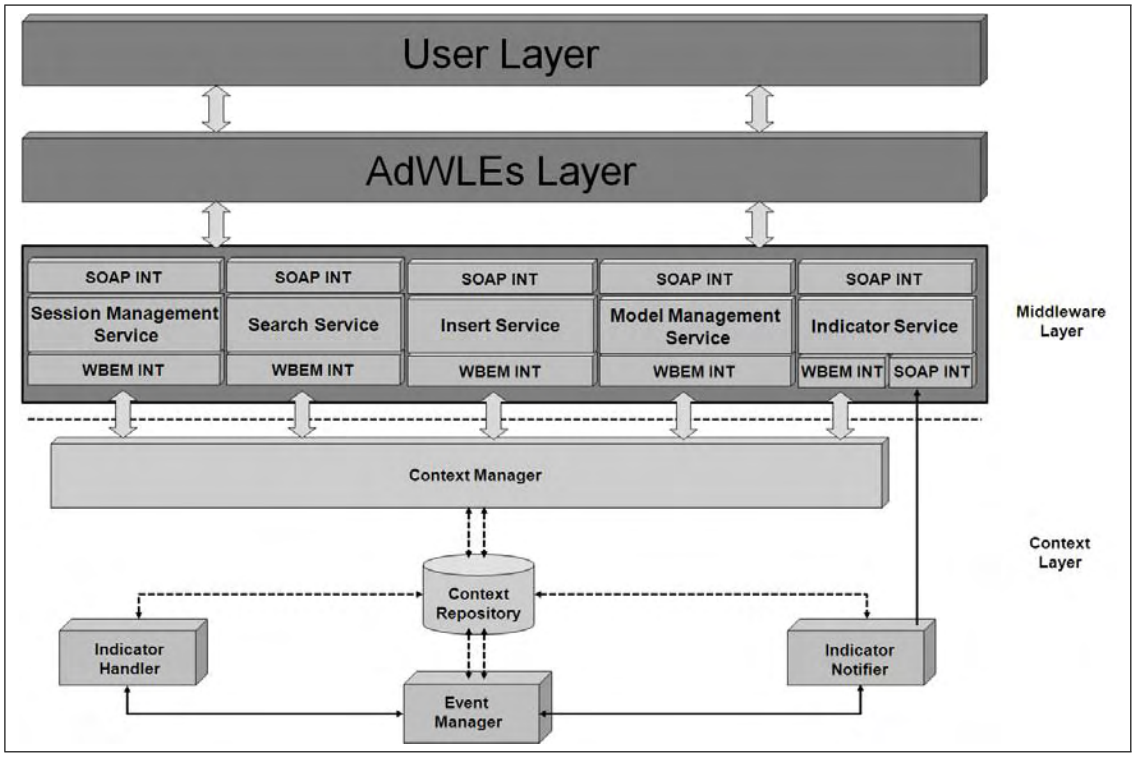


Figure 5.46: Details of the middleware layer.

5.2.2.1 The Session Management Service

This service exposes three methods to ensure the identification of a consumer. Indeed, as shown later in this section, a consumer of the services exposed by the middleware layer is offered the opportunity of setting various parameters according to its preferences. Thus, a session records these parameters, so that the configuration specified by a consumer can be taken into account when any communication occurs.

createAnonymousSession

The method *createAnonymousSession* specified in Table 5.4 creates an anonymous session and returns a session identifier as a string to identify the consumer during further communications.

Method name	**createAnonymousSession**
Return type	String
Fault	METHOD_FAILURE

Table 5.4: createAnonymousSession.

When a consumer invokes this method, some default values are applied to the configuration parameters and stored within the context repository; these values are detailed in Chapter 6.

createSession

The method *createSession* described in Table 5.5 provides the same features as the previous method, but requires credentials (e.g. user, password) to produce a session identifier. The aim of this method is to restrict the use of some services to specific consumers or group of consumers. As an example, treatments operated by the methods of the model management service are critical in the sense that the modeling of context can be compromised and become inconsistent. Therefore, the administrator of the whole framework is responsible for assigning credentials to qualified consumers / groups only, thus preventing (or at least restricting) important issues to occur.

Method name	**createSession**	
Return type	String	
Parameters	*Name*	*Type*
	userID / groupID	String
	password	String
Fault	WRONG_CREDENTIALS METHOD_FAILURE	

Table 5.5: createSession.

destroySession

This method exposed in Table 5.6 is called by a consumer to destroy a session when interactions with the middleware layer are achieved. If this method is not explicitly invoked after communications, the session management service deletes the matching record after a given period of time.

Method name	**destroySession**	
Return type	Void	
Parameters	*Name*	*Type*
	targetSessionID	String
Fault	NO_SUCH_SESSION METHOD_FAILURE	

Table 5.6: destroySession.

5.2.2.2 The Search Service

The search service allows consumers to retrieve context data stored into the repository, and complies with the Simple Query Interface (SQI) specification [SMVA+05]. SQI has been designed to facilitate queries over heterogeneous repositories, with a focus on interoperability: SQI is independent from data stored into repositories, query languages and result formats. One of the advantages of this specification is that a consumer may express queries using a language that best suits its needs, as well as order a result format appropriated to its objectives. The service is then responsible for mapping the query language to the the internal language of the target repository, and formatting the results retrieved from the repository according to the consumer requirements.

Thus, this service allows a consumer to configure queries and results (in terms of query language, result format, number of results, etc.) before sending requests to the context repository. Let us note that each method requires a session identifier as input, so that each configuration parameter is associated with the matching consumer. To complete these process, the search service exposes a set of methods appearing in Table 5.7 and described below. However, each method related to the configuration phase is optional, since the session management service affects default values to each configuration parameter when a session is created.

Phase	Method name
Request configuration	getAvailableQueryLanguages setQueryLanguage getAvailableResultsFormats setResultsFormat setMaxQueryResults setResultsSetSize
Query management	getTotalResultsCount synchronousQuery

Table 5.7: The methods of the search service.

The method *getAvailableQueryLanguages* allows consumers to retrieve the list of available query languages supported by the search service in which they can express their queries.

The method *setQueryLanguage* allows a consumer to control the syntax of the query statement by setting the query language. A fault may occur if the requested query language is not supported by the

search service.

The method *getAvailableResultsFormats* allows consumers to retrieve the list of available results formats in which they can retrieve the results.

The method *setResultsFormat* allows a consumer to control the format of the results returned by the search service. A fault may occur if the required format is not supported by the search service.

The method *setMaxQueryResults* defines the maximum number of results a query will produce. This configuration parameter is useful when the consumer is not interested in receiving all results (e.g. the total number of results is too big). If this number is set to 0 (zero), the consumer does not want to limit the number of results matching with the query. A fault may occur if an invalid number is provided.

The method *setResultsSetSize* specifies the maximum number of results which will be returned by a single results set; a results set is a sub-set of the whole amount of results matching with a given query, and helps to paginate results. A consumer asks for all results when the results set size is set to 0 (zero). A fault may occur if an invalid number is provided.

The method *getTotalResultsCount* returns the total number of results for a given query and may generate a fault if the query statement does not comply with the syntax of the configured query language.

Finally, the method *synchronousQuery* allows to execute a query on the context repository and returns a set of records. It requires, in addition to the session identifier, two arguments: the query statement and an integer indicating the beginning of the results set. Some faults may occur if the query statement does not comply with the syntax of the query language, or if the start result is upper than the total number of results.

Use case: Querying the Context Repository

The UML sequence diagram illustrated in Figure 5.47 is composed of two parts. The upper side of the figure represents interactions which occur when a consumer authenticates itself to the Middleware layer through the session management service. The bottom side of the figure shows the interactions that take place when the consumer sends a query to the search service (in this example, default values apply since no configuration methods have been invoked). The steps required for authentication are the following ones:

1. The consumer invokes the *createAnonymousSession* method of the session management service in order to receive a *SessionID*;

2. The service generates a *SessionID*;

3. The configuration parameters are set to their default values, and the method creates the statement to insert the matching session record (*SessionID* + default configuration parameters) into the context repository;

4. The query is forwarded to the repository manager for execution;

5. The session record is inserted into the context repository;

6. The *SessionID* is returned to the consumer for further reuse.

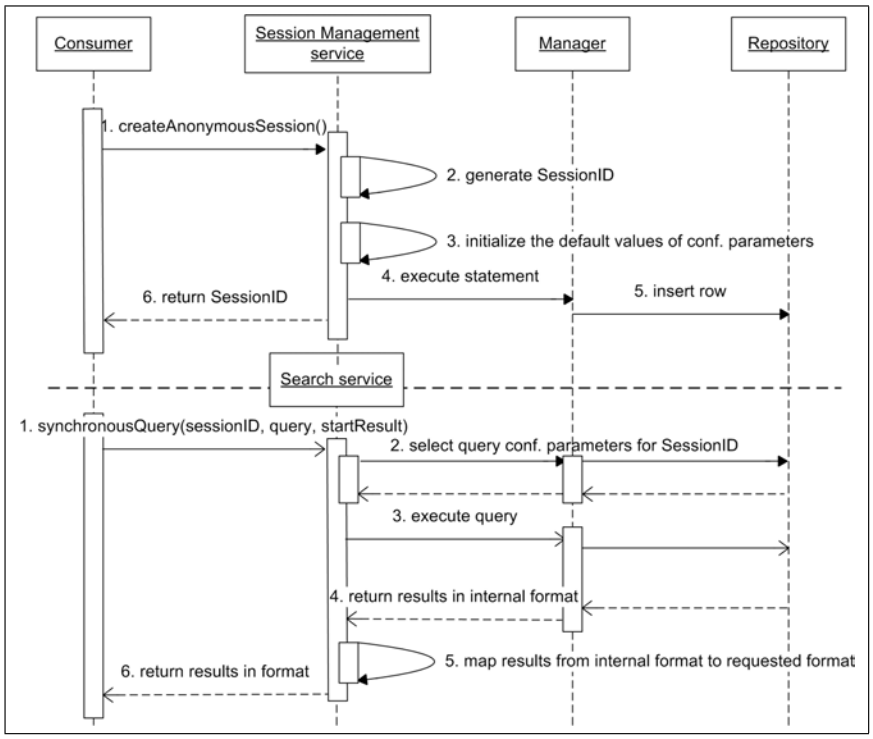


Figure 5.47: Authentication and query process.

Once the consumer gets a session identifier, it is able to call the *synchronousQuery* method in order to query the repository:

1. The consumer sends a request to the *synchronousQuery* method and provides its session identifier, the query statement, and the number of the first result to be returned;

2. The search service sends a query to the repository to retrieve the configuration parameters associated with the given *SessionID*. If no results are found, an exception is returned;

3. The search service re-designs the query in order to consider the configuration parameters (the query language, the maximum number of results, the size of the results set and the position of the first result), and then forwards the query to the manager;

4. The query is processed and the results are returned to the search service according to the internal format of the repository;

5. The results are further formatted from the internal format to the configured result format;

6. The set of results is returned to the consumer to be further processed.

While the search service allows the retrieval of context data, the insert service allows to populate the context repository with heterogeneous data and to semantically validate this information against the structure of the model.

5.2.2.3 The Insert Service

The insert service allows consumer applications to index context data into the repository. Like the search service, it is not specific to a given format of metadata; instead consumers can send data according to the format that suits their needs best. Indeed, this service is compliant with the Simple Publishing Interface (SPI) [TMVA+08] specification dedicated to the indexation of learning resources and metadata into a repository. SPI makes a distinction between the submission of a metadata instance and the submission of a resource (i.e. its content); since the context repository does not store content but metadata only, the insert service comprises three methods: *getAvailableMetadataFormats*, *setMetadataFormat* and *submitMetadata*.

The method *getAvailableMetadataFormats* allows consumers to retrieve the list of available formats accepted by the insert service to insert context metadata.

The method *setMetadataFormat* allows consumers to specify a given context metadata format that will be used during further communications to index data. The format parameter is provided via a URI or predefined values, and then stored within the session record matching with the identifier of the consumer: as the search service, consumers have to create a session through the session management service before invoking any method of the insert service. A fault may occur when the requested format is not supported by the insert service.

The method *submitMetadata* is responsible for (1) checking if the context metadata included into the consumer's request is syntactically and semantically conform to the configured format of the matching session, (2) parsing the record (when the previous step is validated only) and instantiating the matching classes of the context models, and (3) indexing the instances into the repository through the manager. A fault occurs when the context metadata does not comply with the syntax of the metadata format.

Use case: Indexation of a Context Metadata Record

Figure 5.48 represents the UML sequence diagram translating the interactions and treatments that occur when a consumer has to publish context metadata. Here, it is assumed that the consumer has already created a session:

1. The consumer invokes the *setMetadataFormat* method to specify the format of the context metadata to be published;
2. The insert service sets the requested format into the session record;
3. The consumer invokes the *submitMetadata* method in order to send the context metadata;
4. The service parse the data to check if it complies with the format associated to the session (for validation purposes);
5. If the context metadata is valid, the *submitMetadata* method builds the statements that are required to insert the matching instances of our models according to the internal language of the repository;
6. The service communicates with the Manager to forward the internal-based statements;
7. The insert statements are executed by the Manager so that instances are stored within the repository.

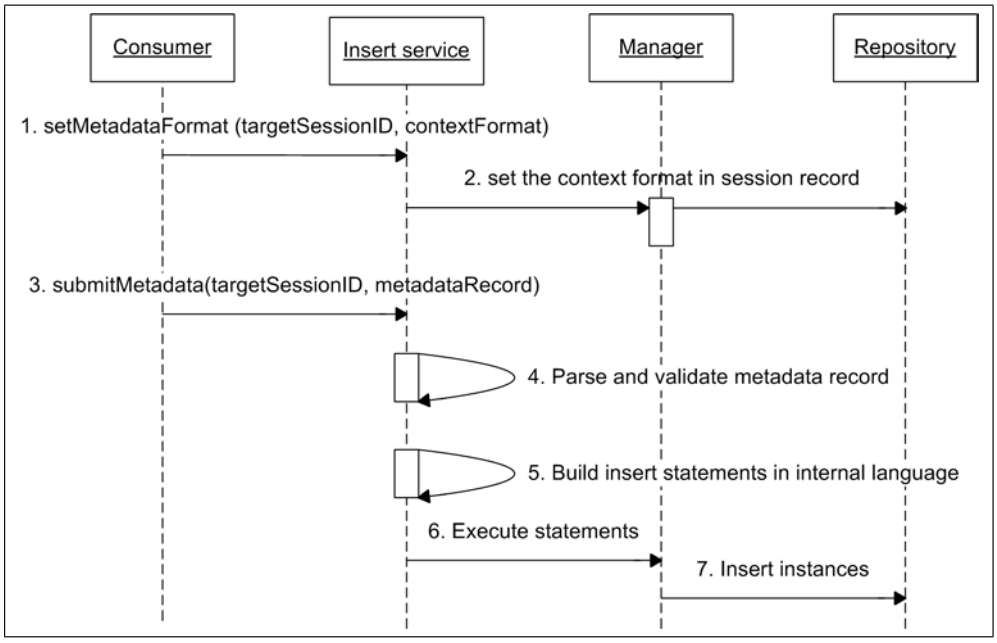


Figure 5.48: Inserting a context metadata record.

When new learning applications have to be observed, they have their own particularities, thus their own context. The next service is designed to extend our context models to consider these specificities.

5.2.2.4 The Model Management Service

The model management service offers a means to extend the generic models when specific contexts must be collected [BVB09]. However, some restrictions apply: one can only define new classes inheriting from the generic user, environment or usage models (TEL classes only can be extended), and one can't modify the specifications of the root models presented in Chapter 4.

If the search and insert services are accessible through anonymous sessions, the model management service is restricted to authenticated consumers (using the *createSession* method of the session management service). Indeed, treatments operated by this service are crucial regarding the consistency of the context models, and experts only should invoke the methods described below. If the *targetSessionID* parameter corresponds to a normal consumers, and not an authenticated one, an unauthorized session exception is raised.

The method *getAvailableClassFormats* (see Table 5.8) allows consumers to retrieve the available formats supported by the model management service to represent a class to be extended or updated.

Method name	**getAvailableClassFormats**	
Return type	String	
Parameters	*Name*	*Type*
	targetSessionID	String
Fault	NO_SUCH_SESSION UNAUTHORIZED_SESSION METHOD_FAILURE	

Table 5.8: getAvailableClassFormats.

The method *setClassFormat* allows a consumer to specify the format in which it will send the class description (see Table 5.9). The *classFormat* parameter is provided via a URI or via predefined values and stored into the session record identified by the *targetSessionID* parameter. A fault may occur when the format provided via the *classFormat* parameter is not supported by the service.

Method name	**setClassFormat**	
Return type	Void	
Parameters	*Name*	*Type*
	targetSessionID	String
	classFormat	String
Fault	NO_SUCH_SESSION CLASS_FORMAT_NOT_SUPPORTED UNAUTHORIZED_SESSION METHOD_FAILURE	

Table 5.9: setClassFormat.

The method *submitClass* allows a consumer to either extend the context models by adding a new class required for some specific observation needs, or to update an existing class. In both cases, the method receives as parameters, besides the *targetSessionID*, a string describing the class to be added/ updated in terms of inheritance, name, data types and constraints.

Method name	**submitClass**	
Return type	Void	
Parameters	*Name*	*Type*
	targetSessionID	String
	classMetadata	String
Fault	NO_SUCH_SESSION CLASS_METADATA_NOT_VALID NO_SUCH_SUPERCLASS NO_SUCH_DATATYPE INVALID_CONSTRAINT UNAUTHORIZED_SESSION METHOD_FAILURE	

Table 5.10: submitClass.

The following faults can occur:

- NO_SUCH_SESSION if the given *targetSessionID* is invalid;
- CLASS_METADATA_NOT_VALID if the *classMetadata* does not comply with the syntax of the format set by the previous method;
- NO_SUCH_SUPERCLASS when the parent class of a new class does not exist into the context models;
- NO_SUCH_DATATYPE when a datatype value does not match to any datatype;
- INVALID_CONSTRAINT if a datatype constraint is not valid (e.g. the length of an attribute is being decreased);
- UNAUTHORIZED_SESSION if the *targetSessionID* parameter does not correspond to an authenticated consumer;
- METHOD_FAILURE if the operation fails for another reason.

While the model management service is designed to manage the raw context model, the indicator service presented next is responsible for the management of inferred context model. Even if the definition of new indicators may be achieved through the model management service, the inferred data require additional methods that are specific to this type of data only.

5.2.2.5 The Indicator Service

The indicator service plays a double role: on the one hand, it allows consumers to define new indicators, subscribe to indicators in order to receive notifications when a fresh value has been calculated, and un-subscribe from one or more indicators. On the other hand, this service accepts requests sent by the indicator notifier and receives as input the definition and value of a newly calculated indicator; it then pushes this information to all subscribers of the matching indicator. To offer these capabilities, the indicator service comprises eight methods detailed below: *getAvailableDefinitionFormats*, *setDefinitionFormat*, *getAvailableAlgorithmLanguages*, *setAlgorithmLanguage*, *defineIndicator*, *getIndicatorsDefinitions*, *subscribeToIndicator* and *unsubscribeFromIndicator*.

The method *getAvailableDefinitionFormats* (see Table 5.11) allows consumers to retrieve the available formats supported by the indicators service to define a new indicator or to receive the list of available indicators definitions.

Method name	**getAvailableDefinitionFormats**	
Return type	String	
Parameters	*Name*	*Type*
	targetSessionID	String
Fault	NO_SUCH_SESSION UNAUTHORIZED_SESSION METHOD_FAILURE	

Table 5.11: getAvailableDefinitionFormats.

The method *setDefinitionFormat* specified in Table 5.12 allows consumers wishing to define a new indicator or to get the list of existing indicators' definitions, to specify the format of these definitions. The *definitionFormat* parameter is provided via a URI or via predefined values, and is stored within the session record associated to the *targetSessionID*. A fault may occur if the requested format is not supported by the service.

Method name	**setDefinitionFormat**	
Return type	Void	
Parameters	*Name*	*Type*
	targetSessionID	String
	definitionFormat	String
Fault	NO_SUCH_SESSION DEFINITION_FORMAT_NOT_SUPPORTED UNAUTHORIZED_SESSION METHOD_FAILURE	

Table 5.12: setDefinitionFormat.

The method *getAvailableAlgorithmLanguages* (see Table 5.13) allows consumers to retrieve the list of available languages supported by the indicators service to define the algorithm which conducts to the calculation of an indicator.

Method name	**getAvailableAlgorithmLanguages**	
Return type	String	
Parameters	*Name*	*Type*
	targetSessionID	String
Fault	NO_SUCH_SESSION UNAUTHORIZED_SESSION METHOD_FAILURE	

Table 5.13: getAvailableAlgorithmLanguages.

The method *setAlgorithmLanguage* allows consumers wishing to define complex indicators to define the language used to express the algorithm which leads to the calculation of the indicator (see Table 5.14). This configuration parameter is stored into the session record associated to the *targetSessionID* parameter, and allows consumers to be independent from a given algorithm language. A fault occurs when the language provided via the *algorithmLanguage* parameter is not supported by the service.

Method name	**setAlgorithmLanguage**	
Return type	Void	
Parameters	*Name*	*Type*
	targetSessionID	String
	algorithmLanguage	String
Fault	NO_SUCH_SESSION ALGORITHM_LANGUAGE_NOT_SUPPORTED UNAUTHORIZED_SESSION METHOD_FAILURE	

Table 5.14: setAlgorithmLanguage.

The method *defineIndicator* detailed in Table 5.15 allows consumers to define a new elementary, arithmetic or complex indicator definition; in other words, it allows to create *instances* of the TEL indicator model specified in the previous chapter; its aim is not to extend this model. The method receives as parameters, in addition to the *targetSessionID*, a string specifying the properties of the class *CIM_BaseMetricDefinition* (see Figure 4.40) and compliant with the definition format, together with the mathematical function or formula or algorithm that leads to the calculation of the indicator value. In case of an algorithm, the consumer has to invoke the method *setAlgorithmLanguage* first in order to specify the language used to express the calculation algorithm. Moreover, since it may be tricky to define new indicators (and more precisely the way the value(s) is calculated), this method is restricted to consumers authenticated through the *createSession* method only. The following faults can occur:

- METADATA_RECORD_NOT_VALID if the *instanceMetadata* does not comply with the syntax of the definition format;
- NO_SUCH_CLASS when the class referenced by the instance does not exist;
- ALGORITHM_NOT_VALID when the algorithm used to calculate the indicator value does not comply with the syntax of the algorithm language;

- UNAUTHORIZED_SESSION if the *targetSessionID* parameter does not correspond to an authenticated consumer.

Method name	**defineIndicator**	
Return type	Void	
Parameters	*Name*	*Type*
	targetSessionID	String
	instanceMetadata	String
Fault	NO_SUCH_SESSION METADATA_RECORD_NOT_VALID NO_SUCH_CLASS ALGORITHM_NOT_VALID UNAUTHORIZED_SESSION METHOD_FAILURE	

Table 5.15: defineIndicator.

When invoked, the method *getIndicatorsDefinitions* specified in Table 5.16 returns a list composed of either:

- The whole set of indicator definitions stored into the repository (i.e. the instances of the class *TEL_IndicatorDefinition* and its sub-classes); in that case, the optional *className* parameter is not provided and each entry of the results list comprises, in addition to the detailed indicator definition, the name of the class(es) on which it applies to;

- Or the indicator definitions that apply to a given entity of the context models only; in that case, the *className* parameter is set to a specific value, and the method exploits the association *CIM_MetricDefForMe* to find out indicators defined for the specific class.

Here, a fault may occur if the *className* parameter does not exist into the raw context models.

Method name	**getIndicatorsDefinitions**	
Return type	String	
Parameters	*Name*	*Type*
	targetSessionID	String
	className	String
Fault	NO_SUCH_SESSION NO_SUCH_CLASS METHOD_FAILURE	

Table 5.16: getIndicatorsDefinitions.

The method *subscribeToIndicator* allows consumers to be notified when an indicator of interest is (re)calculated (see Table 5.17), and thus to subscribe to an indicator. A consumer has to provide three parameters in addition to its session identifier: *indicatorDefinitionID* specifies the indicator of interest, *telInstanceID* represents the TEL context instance on which the indicator value is calculated, and the

listenerURL indicates the location of the listener in charge of receiving notifications when the indicator of interest is (re)calculated. The following faults can occur:

- INDICATOR_DOES_NOT_EXIST when the requested indicator definition does not exist;
- INSTANCE_DOES_NOT_EXIST when the TEL instance does not exist into the repository;
- INVALID_INDICATOR_ME_ASSOCIATION when the indicator definition does not apply to the TEL instance;
- LISTENER_NOT_REACHABLE when the listener is not reachable;

Method name	**subscribeToIndicator**	
Return type	Void	
Parameters	*Name*	*Type*
	targetSessionID	String
	indicatorDefinitionID	String
	telInstanceID	String
	listenerURL	String
Fault	NO_SUCH_SESSION INDICATOR_DOES_NOT_EXIST INSTANCE_DOES_NOT_EXIST INVALID_INDICATOR_ME_ASSOCIATION LISTENER_NOT_REACHABLE METHOD_FAILURE	

Table 5.17: subscribeToIndicator.

The method *unsubscribeFromIndicator* detailed in Table 5.18 is the opposite of the previous one, since it allows consumers to unsubscribe from receiving notifications about an indicator. Three of the four parameters are optional:

- If *listenerURL* is provided, the service will delete all the subscriptions of the current consumer;
- If *indicatorDefinitionID* is provided as well, the service will delete all subscriptions related to the given definition only;
- Furthermore, if *telInstanceID* is provided too, the service will delete the subscription related to the given definition and TEL instance only;
- Finally, if *indicatorDefinitionID* is not provided, the service will delete all subscriptions related to the TEL instance.

Several faults may occur:

- NO_SUCH_INDICATOR_SUBSCRIPTION when no subscriptions exist for the indicator definition;

Method name	**unsubscribeFromIndicator**	
Return type	Void	
Parameters	*Name*	*Type*
	targetSessionID	String
	indicatorDefinitionID	String
	telInstanceID	String
	listenerURL	String
Fault	NO_SUCH_SESSION NO_SUCH_INDICATOR_SUBSCRIPTION NO_SUCH_ME NO_SUCH_INSTANCE LISTENER_NOT_REACHABLE METHOD_FAILURE	

Table 5.18: unsubscribeFromIndicator.

- NO_SUCH_ME when no subscriptions exist for the given indicator definition and TEL instance;
- NO_SUCH_INSTANCE when no subscriptions exist for the TEL instance;
- LISTENER_NOT_REACHABLE when the listener is not reachable.

The set of services of the middleware layer facilitates the manipulation of context metadata by the consumers, both at the class and instance levels, wether they are interested in the raw or inferred contexts. In the following section, we expose the components that should be integrated into the AdWLEs to interact with these services.

5.2.3 The AdWLE Layer

The AdWLE layer comprises tools and applications from which context metadata is collected and/or reused for adaptation purposes. These applications may embed various components illustrated on Figure 5.49 that can be seen as plugins of AdWLEs:

- **A sensor** is in charge of extracting the information defined within our context models from the application. Sensors are consumers of the insert service, where the communication is initiated as soon as an activity is performed by a user. This component also communicates with the user layer (see next section);
- **An adaptive component** (i.e. adaptation engine) interacts with the two adjacent layers to suggest various adaptation mechanisms. It thus retrieves context data through the search service of the middleware layer, and via the user layer (see below).
- **An indicator subscriber/unsubcriber** is able to subscribe/unsubscribe to/from indicators through the indicator service. This component serves the adaptation engine, and may even by integrated into this component;

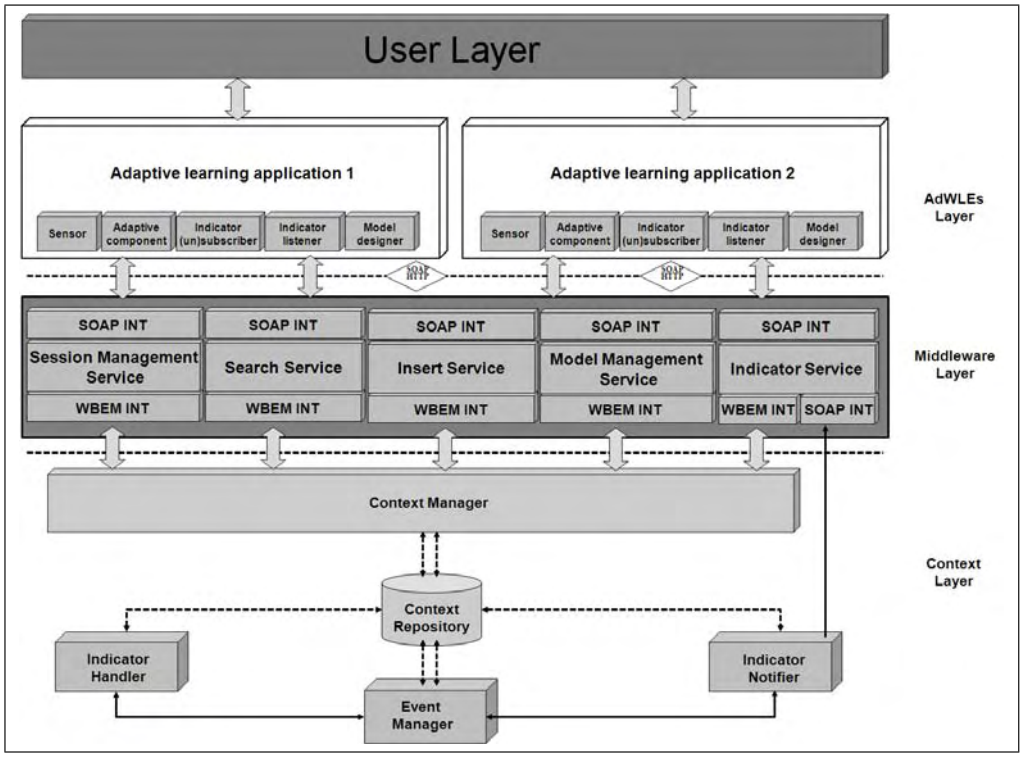


Figure 5.49: Details of the AdWLE layer.

- **An indicator listener** allows applications to receive notifications from the indicator service about the calculations of TEL indicators of interest;
- **A model designer GUI** is intended for various actors of the learning process to express their context information needs. This component exploits both the model management service and the indicator service to extend the raw context model and/or to define new indicators.

If sensors and adaptive components are closely coupled with the hosting learning application and may require a significant development effort, the indicator subscribers and listeners are simple web service clients that may be easily reused from an application to another.

5.2.4 The User Layer

The layers of our architecture presented until now do not take into account privacy of users, and even provides third party applications with an easy access to confidential data: the context repository stores very sensitive data (e.g. the user context), and the middleware layer offers an access to this data at a large scale.

Within the security area, a general principle is to distribute sensitive data rather than centralize it within a single information system. Thus, the risk that someone gets an overview of the whole set of

information is very low [BDF+05]. We adopted this principle in our architecture as well: the separation of the system storing personal data, or sensitive data, and the system storing less sensitive data which does not present any danger [BVB11c]. The resulting architecture is illustrated in Figure 5.50: the user context which can lead to revealing somebody's identity resides on the local device of the user, whereas the environment and usage contexts are recorded on the repository of the context layer. Therefore, when a context metadata record is collected from an AdWLE, the sensor splits information into two distinct parts: sensitive information (i.e. the user context) is indexed into the local device of the user, and non sensitive data (i.e. the environment and usage context) is forwarded to the insert service to be further stored into the context repository; the whole scenario is depicted in Figure 5.51, where an additional step has been added to Figure 5.48. Also, the adaptive component has to query both the central repository (through the search service) and the local storage system of the user to retrieve the whole set of context metadata. Then a question arises: how to identify the user within the central repository, since no user data is provided by the search service?

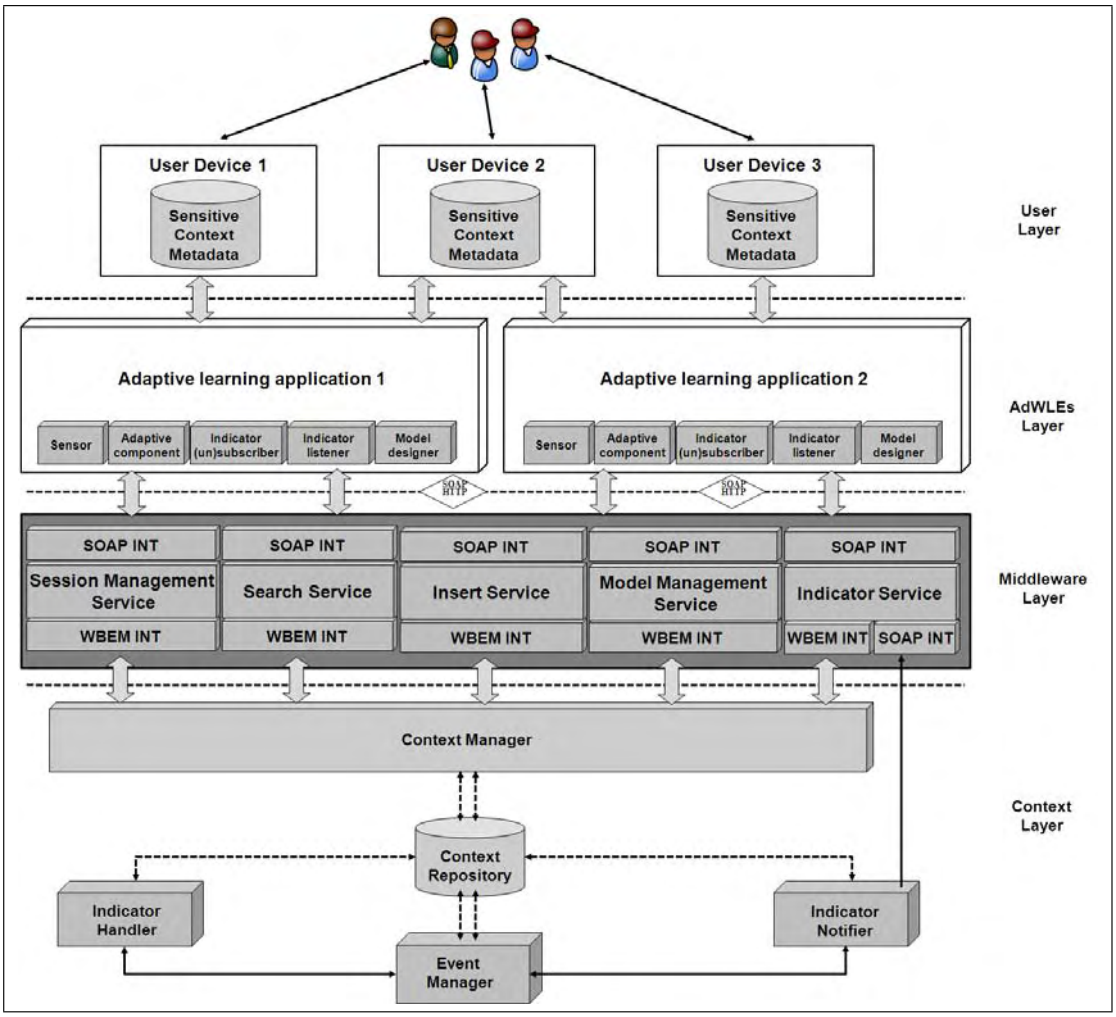


Figure 5.50: Details of the user layer.

Chapter 2 highlighted the benefits of pseudonymous authentication when an application or system has to identify a user without revealing or knowing his/her identity. Our proposal stands on this technique, and consists in storing into both the local and central repositories the same identifier for a given user; thus, the class *CIM_Identity* only is instanciated within the context repository, and the single *InstanceID* property is valued.

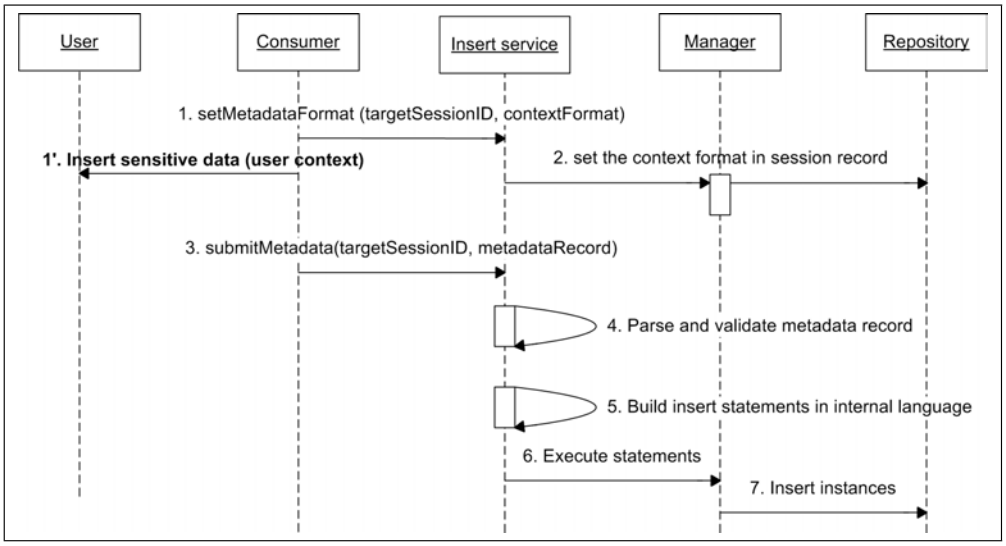


Figure 5.51: Taking into account users' privacy when inserting context.

This organisational principle prevents the context repository of being subject to privacy concerns, since it does not contain any sensitive data from which users' identity can be revealed. On the other hand, we consider that the local storage system of the user device has to provide its own security mechanisms to prevent unauthorized access to sensitive data; two alternatives are suggested in the next chapter to implement such a system. Moreover, the features/guidelines that the components of the architecture have to address to ensure privacy of users are the following ones:

- Sensors have to notice users that their context is collected (**notice principle**), but also inform them about the purpose(s) of this process (**purpose principle**). Since the scope of our framework is to widely share and reuse context data, sensors have to warn users that the collected information can be accessed by anyone, for any learning purpose; however, users must be aware that their identity cannot be revealed through the collected data. Finally, sensors have to expose to users, in an intelligible form, the nature of the data that are going to be collected (**consent principles**);
- Adaptive components have to ensure that the sensitive data processed by the AdWLE is protected against unlawful destruction or accidental loss, alteration, unauthorized disclosure or access (**security principle**); since the central repository does not store sensitive data, it doesn't have to comply with this principle. Moreover, as they access sensitive data, adaptive components must not forward the personal data to any third party (**disclosure principle**);

- Transactions that occur between the sensors/adaptive components and the user device encapsulate sensitive data and must be secured to prevent an attacker to intercept the data;
- Local storage systems responsible for the management of sensitive data have to allow users to rectify and erase their personal data (**access principle**).

5.3 Conclusions

In this chapter we have presented our architecture for collecting, storing and sharing contextual data coming from heterogeneous adaptive web-based learning environments. It is inspired from the DMTF specifications, and is composed of four layers.

The **context layer** comprises a repository hosting the classes and instances of the context models presented in Chapter 4, and a manager responsible for its management; the repository contains the whole set of classes describing the user context, but one class only (*CIM_Identity*) is instantiated in order to prevent the identity of users to be revealed. Besides these, this layer embeds some components dedicated to the management of TEL indicators: they ensure automatic calculation and delivery of indicators to applications interested in this data. Therefore, this layer acts as the storage/management layer of the six-layered architecture described in Chapter 3. As new context information is captured and new applications are observed, the amount of data may rapidly increase: a scalable framework must implement this layer, as shown in the next chapter.

The **intermediate layer** provides a toolbox of web services based on standardized interfaces in order to promote the share and reuse of contextual metadata, and represents both the query and reception layers. Indeed, a search service based on the Simple Query Interface is exposed to query the repository, whereas an insert service based on the Simple Publishing Interface is responsible for indexation of context metadata into the context layer. Both services are neutral in terms of context formats and query languages, thus bringing flexibility and usability: consumers can perform requests to the search service using any query language/result format combination, and new context metadata to index does not have to comply with a specific format. Mappings of formats and query languages towards the internal specificities of the context layer are ensured by distinct wrappers included into the services. However, these mappings may be time consuming for heavy requests, thus efficiency issues may appear. In this case, context metadata may convey to Big Data challenges. Moreover, two additional services make it easy to extend the existing raw models and define/subscribe/unsubscribe new indicators, both of them being independent from any specific format as well.

The **AdWLE layer** represents systems and applications exploited by users, and acts as the last three layers of the six-layered architecture: sensors are responsible for gathering context metadata, adaptive components embed the adaptation logic and reuse the context metadata as input, whereas AdWLEs

represent the application itself. This layer is highly flexible, since each component is independent from the others (a WLE may embed a sensor or an adaptive component only, or both). It is highly open as well: to integrate new applications, the context models specific to this system can be easily designed using the model designer GUI, and the matching sensors can be built on the basis of existing ones (the creation of the metadata record translating the context to be gathered is the single process specific to the target application). Finally, adaptive systems may embed indicator (un)subscriber and listener to benefit from up-to-date values of indicators of interests and process relevant adaptation techniques.

The **user layer** is represented by the user device. We have introduced this layer to handle privacy issues related to the collection, processing and storage of personal data. Therefore, this layer stores the user context only, and communicates with the components of the AdWLE layer to allow storage and retrieval of users profile.

Finally, we have introduced a pseudonymous authentication process to uniquely identify a user on both the context and the user layers, together with a set of requirements and guidelines that guarantees the privacy of users.

The next chapter describes some technologies that may be used to implement our global framework, together with some formats that may be used to store and exchange context metadata.

PART III - Implementation

Chapter 6 - Adopted Technologies and Formats

Contents

In this chapter we present the technologies and tools that implement the four layers of the architecture depicted in the previous chapter. Since we claim the independence of the middleware layer regarding query languages and context metadata format, this chapter also describes the adopted proposals.

6.1 The Context Layer

The context layer is conform to the WBEM specifications, and includes two main components: the repository stores the classes and instances of our models that are created, updated or deleted by the manager.

6.1.1 Selection of a Context Repository

A wide variety of WBEM implementations is currently available, not only as stand-alone applications (e.g. WBEM Solutions[1], OpenPegasus[2], OpenWBEM[3]) but also as embedded components of operating systems dedicated to their management (e.g. Windows Management Instrumentation[4], Apple™ Remote Desktop[5]). Among these tools, we focused on OpenPegasus, an open-source software written in C++.

6.1.1.1 OpenPegasus - A Management Application

OpenPegasus is an open-source implementation of the DMTF CIM and WBEM standards developed by the OpenGroup and designed to be portable and highly modular. It is coded in C++ to effectively map the CIM object concepts to a programming model, while offering speed and efficiency that characterize compiled languages. Among other features, OpenPegasus includes a command-line interpreter (CLI) client to perform general CIM operations over the classes and instances of the repository (e.g. enumeration of classes and instances, execution of queries, etc.), and offers the possibility to choose between an in-memory, binary or SQLite[6] database repository. OpenPegasus adopts the MOF language to support CIM classes and instances, that is the native language elaborated by the DMTF.

Considering the popularity of OpenPegasus among industrials, we have chosen this WBEM implementation as the context layer for the first version of our framework, and manually produced the MOF files matching with our context models. As the number of context metadata instances grew rapidly, the time required by OpenPegasus to retrieve specific information into the repository was becoming more and more important, making the framework almost unusable. OpenPegasus is a very efficient software when it is used for its primary purpose: to manage applications, networks, systems or services. Indeed, the amount of managed resources (i.e. the number of instances within the CIM repository) is not that large, and can vary from a few dozens (in case of a small company) to a few hundreds or thousands (in case of medium or big companies). OpenPegasus works fine when the number of instances remains

[1] WBEM Solutions - http://www.wbemsolutions.com/
[2] OpenPegasus - http://www.openpegasus.org/
[3] OpenWBEM - http://www.openwbem.org/
[4] Windows Management Instrumentation - http://msdn.microsoft.com/en-us/library/ms811553.aspx
[5] Apple Remote Desktop - http://www.apple.com/remotedesktop/
[6] SQLite - http://www.sqlite.org/

low, but fails when the number of instances becomes important (i.e. gigabytes of data). When observing precise interactions between users and tools of a learning environment, the number of instances stored into the CIM repository grows fast to reach thousands of units for a single user, or hundred of thousands or event millions for many users interacting with several applications.

Thus, OpenPegasus failed to act as a real database management system offering query optimisation features such as cache memory or indexes. As a consequence, we looked at other alternatives to implement the context environment.

6.1.1.2 eXist - An Open-source Native XML Database

eXist-db is an open source database management system (DBMS) built using XML technologies. It stores XML data and features efficient, index-based XQuery processing. The database is completely written in Java and may be deployed in various ways, either running as a stand-alone server process, or inside a servlet-engine, or even directly embedded into an application.

eXist provides storage of XML documents organized into hierarchical collections, and exposes an enhanced indexing schema that supports quick identification of structural relationships between nodes. eXist also provides a number of extensions from standard XPath or XSLT to fulltext queries, including keyword searches, queries on proximity of search terms or regular expressions. For developers, access through HTTP, XML-RPC, SOAP and WebDAV are available, and Java applications may use the XML:DB API.

The database suits applications dealing with small to large collections of XML documents which are occasionally updated [Mei06]. Database indexes are used extensively by eXist to facilitate efficient querying of the database. This is accomplished both by system-generated and user-configured database indexes.

Therefore, to evaluate this DBMS, we had to represent our context models using the XML format. Since the DMTF defined the xmlCIM format to represent CIM classes and instances through the XML language, this process was almost straightforward: one class or instance of our models was mapped to an XML node.

6.1.1.3 Oracle Object-relational Database

An object-relational database (ORD), or object-relational database management system (ORDBMS), is a traditional DBMS characterized by an object-oriented database model: objects, classes and inheritance are natively supported by both the database schemas and the query language.

Oracle object-relational (O-R) database implements the object-type model as an extension of the relational model, while continuing to support standard relational database functionalities such as queries,

fast commits, backup and recovery, scalable connectivity, row-level locking, read consistency, etc. Various programmatic interfaces and languages including SQL, PL/SQL, Java, OO4O or C# have been enhanced with extensions to support Oracle objects. Just like eXist, Oracle includes various mechanisms to improve the speed of SQL queries. Taking advantage of the low cost of disk storage, Oracle includes many new indexing algorithms that dramatically increase the speed required to process queries. Oracle also uses indexes to avoid the need for large-table/full-table scans and disk sorts. The result is an object-relational model that offers intuitiveness and economy of an object interface, while preserving the high concurrency and throughput of a relational database.

Oracle defines a class as an object type composed of attributes and methods. Attributes hold the data about an object while methods are procedures or functions which apply on attributes' values. Figure 6.52 shows the mapping of the MOF *TEL_Resource* class definition to SQL. The syntax is very similar, even if few differences appear: (1) a class in Oracle O-R is seen as an object type, (2) an abstract class is defined as "NOT INSTANTIABLE", and (3) a class which allows sub-typing is declared as "NOT FINAL". Moreover, Oracle O-R distinguishes the storage location of the class definitions and the associated instances: the user has to specify where instances must be stored. In the case of Figure 6.52 , we created a table of resources which contains the instances of all learning resources sub-types. Therefore, in order to map our models to Oracle O-R, we created one object table for each class of the generic models, each object table containing the instances of all sub-classes of the root class.

```
[Abstract]
class TEL_Resource: CIM_SystemResource {
    [Key, MaxLen(400)]
    string Identifier;
    [MaxLen(400)]
    string ElementName;
    [MaxLen(4000)]
    string Description;
    [Required]
    datetime CreationDate;
    datetime DeletionDate;
    string Language;
};
```

⇨

```
create TYPE TEL_Resource AS CIM_SystemResource (
    Identifier VARCHAR2(400),
    ElementName VARCHAR2(400),
    Description VARCHAR2(4000),
    CreationDate TIMESTAMP,
    DeletionDate TIMESTAMP,
    Language VARCHAR(100)
) NOT INSTANTIABLE NOT FINAL;

CREATE TABLE TEL_Resource_table OF TEL_Resource
    (Identifier PRIMARY KEY, CreationDate not null);
```

Figure 6.52: Mapping TEL_Resource.mof to SQL.

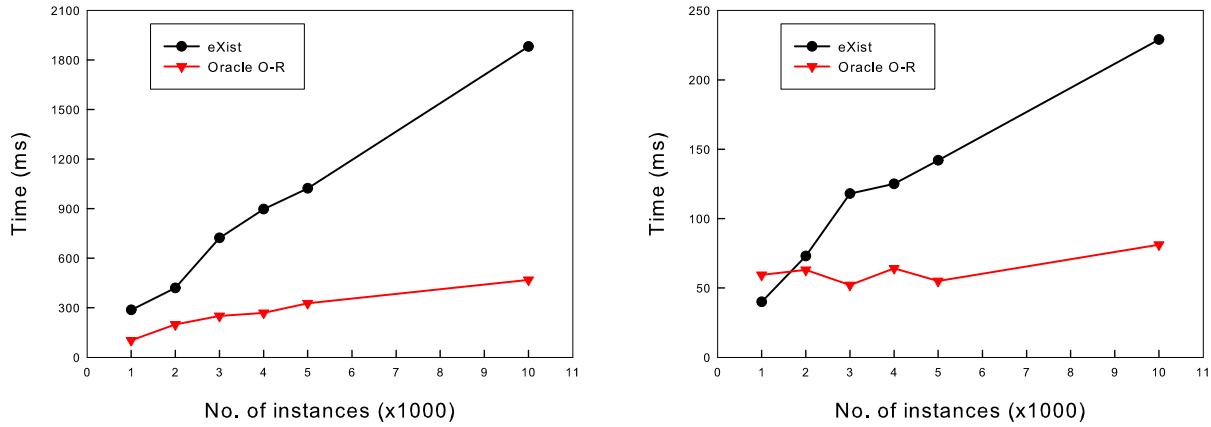
The next section presents some comparative tests between eXist and Oracle O-R related to execution time performances.

6.1.1.4 Execution Time Performance Evaluation

This section exposes some performance evaluation results concerning the time required by eXist and Oracle O-R to process queries. These tests are composed of two experiments: the first one aims at identifying the most scalable software (in terms of number of instances) when simple queries are processed, whereas the second experiment compares the time required to handle complex queries (simple

and double joins) over a fixed set of instances. Each experiment was leaded on a laptop featuring a Mobile Intel Core 2 Duo U7700 CPU, dual core, each core running at 1333 MHz, 2GB DDR2 RAM memory, and running the Kubuntu 9.10 operating system.

The biggest set of data used for the first experiment comprised 10000 instances of the class *TEL_Courseware*, and the query consisted in retrieving all attributes of a random course instance. To draw the curve translating the time required to process the query according to the number of instances, we gradually increased this number from 1000 to 5000 in intervals of 1000, and from 5000 to 10000 in one interval. The tests have been performed with and without indexes for both software, and the results appear in Figure 6.53.



(a) Retrieving one instance (no index)

(b) Retrieving one instance (with index)

Figure 6.53: Response time comparative charts.

eXist has the most emphasized growth without indexes, as it retrieves one instance among 10000 in 2 seconds (see Figure 6.53). Even if there is a small fluctuation around 2000 instances, eXist is characterized by a linear growth. Oracle O-R performs best, one instance among 10000 being retrieved in 0.5 second. Moreover, the Oracle O-R execution time increases very smoothly as the number of instances grows, even with a 5000 units interval.

Figure 6.53b presents the same experimentation, but indexes were built within the two software. The range index was applied to eXist to compare the *Identifier* attribute to a given value, whereas Oracle O-R was set up with the standard B-tree index over the same attribute. As shown on Figure 6.53b, a drastic diminution of the execution time occurs when indexes apply. For 10000 instances, eXist dropped down from 2 seconds to 229 ms, while the time required by Oracle O-R decreased from 500 ms to 81.2 ms. eXist execution time still presents a linear increase according to the number of instances, whereas Oracle O-R keeps a constant average value of 55 ms to process 1000 to 5000 instances, and a smooth increase to 81.2 ms when handling 10000 instances. Even if eXist performs better than Oracle O-R to execute the query over 1000 instances, the later is best suited for our domain area since the number of

instances to handle may become very important.

The second evaluation of these software focus on more complex queries involving simple and double join that are required to retrieve any information from our context models.

One Join Execution Time Evaluation

Here the query is designed to retrieve all instances of a given type of resource (e.g. courseware) that a specific user interacted with. The data set used to conduct the evaluation was composed of 5000 *TEL_IdentityOnResource* instances and 700 instances of *TEL_Resource*, where 430 of them instanciated the class *TEL_Courseware* depicted in the next chapter.

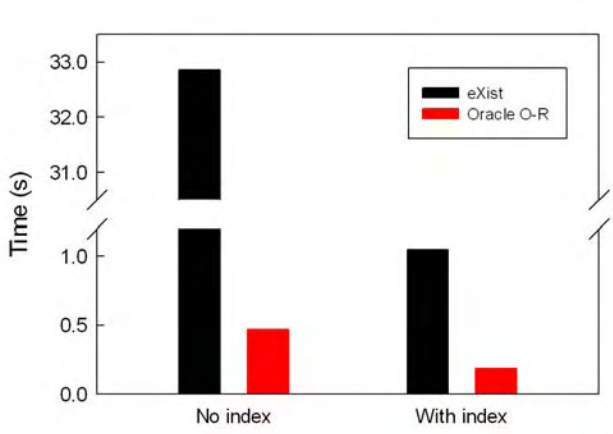


Figure 6.54: Execution time for a one join query.

Again, the experiment was set up with and without indexes (see Figure 6.54). In both cases, it is clear that Oracle performs much better than eXist: 0.47 second versus 32.86 seconds to process the query without indexes, and 187 ms versus 1050 ms when indexes are set up. Oracle O-R performs about six times faster than eXist.

Two Joins Execution Time Evaluation

In addition to searching for all instances of a given resource sub-class (i.e. *TEL_Courseware*) that a user interacted with, this evaluation aims at retrieving, for each returned resource, the instances of a precise activity (i.e. consultation). Here two join operations are required: one between the *TEL_IdentityOnResource* and *TEL_DependencyResourceActivity* classes, and the other between the later and the class *TEL_HasConsulted* (this class is detailed in the next chapter).

Figure 6.55 shows the execution time with and without indexes for both software. Just like the previous evaluation, eXist's response time is huge compared to Oracle: without indexes, eXist required more

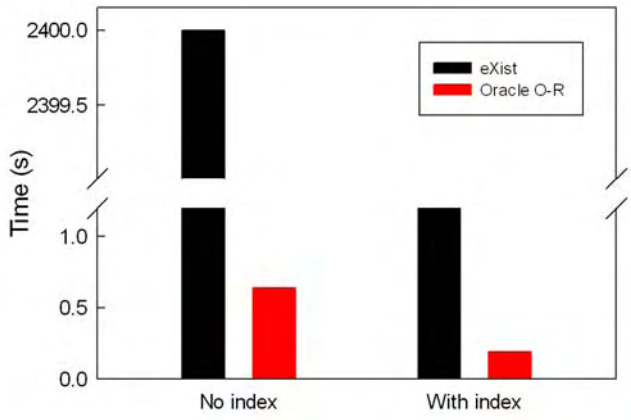


Figure 6.55: Two join response time charts.

than 40 minutes to execute the query while Oracle returned the data after 640 milliseconds.

When using indexes, eXist's response time considerably dropped down to 6.517 seconds, but is still far from the performance of Oracle that required 0.189 second only to execute the query. Let us note the scalability of Oracle which performs two join operations as fast as one join query.

Oracle O-R is a real DBMS, with multiple query optimisation features (such as cache and indexes) and performs much better than eXist in all cases, making it the best candidate for implementing our context repository. Furthermore, the internal mechanism of accessing (referred) objects makes it very easy to write otherwise complex join queries. Finally, Oracle O-R is fully compatible with the CIM model specifications, thus offering a one-to-one mapping of our object-oriented context models without losing semantics.

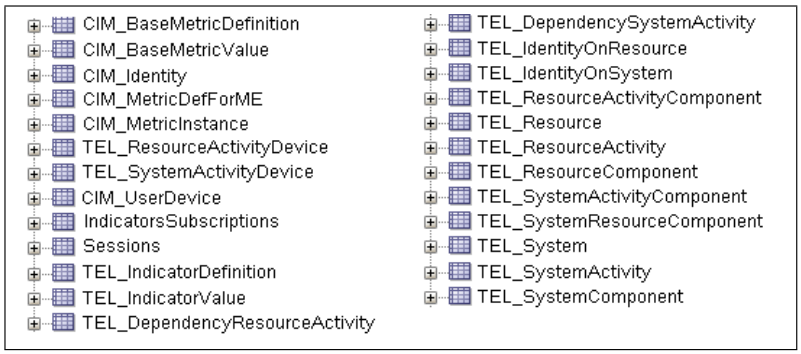


Figure 6.56: The Oracle tables matching with our context models.

Figure 6.56 presents the list of Oracle tables matching with the context models described in Chapter 4, where one table matches with one class of the generic models. In case of an abstract class, the matching table stores instances of all sub-classes; as an example, the table *TEL_Resource* contains instances of all sub-classes of the class *TEL_Resource*. Beside the context tables, two relational tables have been

created to store the session records (table Sessions) and the list of applications that subscribed to one or several indicators (table IndicatorsSubscriptions).

6.1.2 The Indicator Mechanism

The mechanism of indications (or notifications) is implemented in Oracle through the Database Change Notification package (DCN) which allows objects referenced by specific queries to be associated with a callback handler procedure to perform internal or external processings.

The **event manager** belongs to the DCN package and ensures the management of two types of notifications: one is sent to the indicator handler when a new *CIM_MetricDefForME* instance is created, and the other informs the indicator notifier about the creation or update of a *TEL_IndicatorValue* instance.

The **indicator handler** has been developed using the DCN package as well, and is implemented as a stored procedure. Once a notification from the event manager arises, it calculates the indicator value according to the indicator definition: if the indicator has to be calculated periodically (GatheringType=3), the indicator handler re-calculates and updates its value at each *SampleInterval* seconds. It also ensures the creation/modification of the appropriate instances.

Finally, the **indicator notifier** is implemented through the DBMS_ALERT package that offers the opportunity to perform actions outside the DBMS when a specific event occurs. We configured this component to send requests to the indicator service in order to notify the indicators subscribers.

6.2 The Middleware Layer

The services of the middleware layer have been implemented as SOAP web services, and developed using the Java programming language (and more precisely the Java API for XML Web Services java library). In this section we do not focus on the treatment operated by each service; instead we discuss the various input and output formats.

6.2.1 The Session Management Service

As mentioned before, sessions are managed through the *Sessions* table within the Oracle database. Each row contains the configuration parameters used by consumers of the services of the middleware layer to perform actions on the context repository, and set during the configuration phase:

- The query language, the result format, the maximum number of results (100 as default value) and the results set size (25 as default value) are mandatory to send a request to the search service;

- The context metadata format used to insert context records into the repository are required by the insert service;
- The model format used to extend the context models must be set up to invoke the model management service;
- The instance format and the algorithm language (in case of a complex indicator) are used to define new indicators (and required by the indicator service).

The default formats we implemented, together with other alternatives, are detailed next.

6.2.2 The Search Service

The Query Language

The default value is the SQL query language. The main reason for choosing SQL as default query language is to be flexible as much as possible while allowing consumers to express their queries using a standardized and well-known query language. Figure 6.57 gives the SQL request that should be built to query all resources, together with the matching activities, that the user identified by MD5 string has interacted with.

```
SELECT p.Antecedent.Dependent,  p.Dependent

FROM TEL_DEPENDENCYRESOURCEACTIVITY p

WHERE  p.Antecedent.Antecedent.InstanceID='b1a4b2518dbbdd47dd4a713d5cd1df94'
```

Figure 6.57: An example of SQL request.

The Result Format

Two result formats are available: SQL2XML and SQL2xmlCIM. SQL2XML implements the one-to-one mapping of a SQL record to the matching XML nodes. Indeed, each row returned by the SQL query is mapped to a XML node, and the result is returned to the consumer as an XML string containing all nodes. This is the default format we implemented: even if this format is not standardized (consumers have to discover the structure of results by themselves), it suggests a row (SQL) level of semantics and is able to represent heterogeneous classes of our models into a single set of results. Appendix C gives an extract of the set of results returned by the query exposed above.

The second available format implemented by the search service is SQL2xmlCIM. Here, results are returned according to the standardized xmlCIM format [Dav02]. This specification has been estab-

lished to express a CIM class or instance using the XML language: a root xmlCIM node represents a class or an instance where sub-elements describe the qualifiers and attributes of the class/instance (see Figure 6.58). Therefore, it is characterised by the lowest level of semantics, since each cell of a SQL row is mapped to an xmlCIM node. Even if consumers know from the very beginning the format that is going to be returned (an enumeration of XML instances), results can hardly be interpreted when relations between nodes appear.

Thus, depending on the request sent to the search service, consumers have to set the result format parameter to the value that best suits their needs: xmlCIM when they search for instances of a given class (e.g. the resources integrated into a courseware) and SQL2XML when a more complex results set is expected (e.g. the activities performed by the users enrolled into a given a courseware).

6.2.3 The Insert Service

The *submitMetadata* method requires, as main parameter, the context metadata to be indexed into the repository. To express this information, two formats must be considered.

From our point of view, a context metadata in its simplest expression is a tuple {user, learning resource/system, activity on resource/system} where the required elements depend on the target entity on which the activity is performed. Indeed, when a user performs an activity on a resource, the mandatory context metadata comprises the tuple {user, resource, activity, system} to give details about the system hosting the resource, whereas in case of an activity performed over a system, the mandatory set of information is limited to the tuple {user, system, activity}. Furthermore, a resource or system entity may be characterized by various usage and environment contexts. As an example, a context metadata record may translate the fact that a learning object has been integrated into a courseware deployed on a learning management system, or that the learning object has been indexed into a learning object repository; there is a need to precisely express the contexts of the entity being observed.

xmlCIM could be used to represent the context metadata to index, but the consistency of data (e.g. relationships between various elements) cannot (or hardly) be automatically ensured since no semantic constraints are expressed through this format. Instead, the sensor designer is responsible for ensuring himself this consistency; this process can't be set up within a large scale framework.

The second way to represent the context metadata, the one we adopted, consists in imposing a given structure to an XML string according to an XSD schema. XSD schema allows to define the structure and the data type of a given XML document. To meet the requirements of our framework, we designed a schema[1] to check the validity of a given metadata record according to our context models: this schema (a part of it being detailed in Appendix B) is built upon the structure of our models (i.e.

[1] http://osiris.ups-tlse.fr/cm/CMSchema.xsd

relationships between context elements) and dynamically updated each time a modification is applied to the context models through the model management service.

The XML structure of a context metadata record is not flexible, since the validity against the XSD schema ensures the semantic constraints defined within the context models but also the data types of the various attributes and properties. Our dynamic context metadata format brings a series of advantages:

- **One format per observed entity.** Each entity to be observed (i.e. learning resource or system) has its own context metadata structure, as designed within the context models. Thus the sensor designer may refine and express exactly the situation in which an entity is observed.
- **Adaptable and up-to-date.** When the models are updated (e.g. a class, relationship or property is added), the model management service automatically updates the XML schema according to the modification(s).
- **Valid context constraints.** The context metadata structure of each entity complies with our context models by considering the relationships between entities. The data types and possible values of a property are restricted to the ones defined into the models.
- **Different usage contexts for an entity.** When building the context metadata of a given resource or system, the sensor designer can choose the most appropriate context.
- **Gathers as much as possible.** By imposing a specific structure, sensors have to indicate as many information as available from the environment which is observed.

The inconvenient of this structure is that it is proprietary, and consumers have to understand a rather complex XML schema.

6.2.4 The Model Management Service

The model management service exposes the *submitClass* method to extend and modify the existing context models. This method requires a string representing the description of the class(es) to be extended/modified as input parameter, expressed according to a given format. To date, the single available format to represent a class is xmlCIM. Indeed, as mentioned earlier, this standard is perfectly adapted to express a single CIM class or instance as an XML node. Figure 6.58 gives an example to extend the *TEL_Resource* class using this format.

Each time the context models are modified, the model management service updates the XML schema according to the modifications so that sensors can index context metadata that is compliant with the updated models.

```
<CLASS NAME="TEL_Courseware" SUPERCLASS="TEL_RESOURCE">
    <QUALIFIER NAME="Description" TYPE="string" OVERRIDABLE="false" TOSUBCLASS="false">
        <VALUE>
            This class represents a course.
        </VALUE>
    </QUALIFIER>
    <PROPERTY NAME="Category" TYPE="string">
        <QUALIFIER NAME="Required" TYPE="boolean" OVERRIDABLE="false" TOSUBCLASS="false">
            <VALUE>
                TRUE
            </VALUE>
        </QUALIFIER>
        <QUALIFIER NAME="Description" TYPE="string" OVERRIDABLE="false" TOSUBCLASS="false">
            <VALUE>
                This property specifies the category of the course from
                the educational point of view
            </VALUE>
        </QUALIFIER>
    </PROPERTY>
</CLASS>
```

Figure 6.58: Extending the *TEL_Resource* class using the xmlCIM format.

6.2.5 The Indicator Service

The xmlCIM format is used as well to specify the definition of a new indicator, but consumers have to express instances instead of classes. Moreover, the SQL language describes the algorithm to calculate the value of complex indicators. It would be valuable to provide consumers with the possibility of defining algorithms using pseudocode, but no mappings have been set up to date.

This service exploits the *IndicatorsSubscriptions* table holding the list of applications that subscribed to one or several indicators. A record of this table describes the following information: the location of the listener integrated into the learning application of the AdWLE Layer interested in receiving notifications when an indicator of interest is calculated, the identifier of the indicator definition, and the optional identifier of the managed element for which the indicator has to be calculated. When the notifier of the context layer informs the indicator service about the update of an indicator value, the later browses the table to identify applications interested in this indicator, and then notifies the matching listeners.

6.3 The AdWLE Layer

The systems of the AdWLE layer which have been tracked to date are Moodle, the ARIADNE Finder (a search engine dedicated to education that provides access to learning resources worldwide), and a web-based quiz tool.

All these open source tools integrate a specific sensor developed according to the native language of the target application: PHP for Moodle and the quiz tool, and Javascript for the ARIADNE Finder. The implementation of sensors comprises the code to call both the *createAnonymous* method of the session management service to get an identifier (about twenty lines of code) and the *submitMetadata*

method of the insert service to store non-sensitive data into the context layer (about one hundred lines of code: approximatively seventy lines to generate the context metadata record, and a few lines of code to invoke the remote method). In addition, the sensors integrate the code to update the user profile on the device of the user (see further). The detailed context metadata collected by these sensors are exposed in the next chapter.

An adaptive component has been integrated into the ARIADNE Finder to recommend learning objects to the user according to his/her current objectives; both the component and the recommendation process are detailed in Chapter 8.

An indicator listener has been introduced into Moodle as a web service for validation purposes only, since the GUI has not been implemented yet. This listener receives updated values of a single indicator (the proportion of actions detailed in the next chapter), where the matching subscription has been operated through the indicator service.

Finally, due to the complexity of the task but also to the lack of time, no model designers have been developed; discussions about this component appear in the last chapter of this document.

6.4 The User Layer

As mentioned in Chapter 5, the user environment has to ensure the storage of sensitive data, that is the user profile. We suggest here two methods to achieve this task: one implements the DMTF architecture, and the other is based on Internet technologies. Advantages and drawbacks of these approaches are also discussed.

6.4.1 Storing Sensitive Data into a Local WBEM System

As mentioned before, the network and system management approach of the DMTF is natively integrated into most nowadays operating systems to supervise software and hardware components installed on a local machine. Since CIM is used as a basis to model the management knowledge, our user context model can be easily integrated into any local WBEM implementation, and the responsability of ensuring security and access to sensitive data is delegated to the hosting operating system.

We conducted some experiments with Windows Management Instrumentation (WMI) as the local implementation of WBEM (access to WMI is possible through ActiveX objects), and the ARIADNE Finder as the AdWLE. The integrated sensor, fully written in Javascript, interacts with the services of the middleware layer on one hand, and with WMI through an API on the other hand. Figure 6.59 shows an example of a Javascript function to enumerate the name of all user accounts registered on a Windows computer; the class *Win32_UserAccount* is specific to Microsoft™ and matches with the class

CIM_Identity of the CIM model. Thus, our user model has been introduced into the WMI repository using the operation *createClass*.

```
function GetUserNames(){
    var loc = new ActiveXObject("WbemScripting.SWbemLocator");
    var svc = loc.ConnectServer(".", "root\\cimv2");
    coll = svc.ExecQuery("select * from Win32_UserAccount");
    var items = new Enumerator(coll);
    while (!items.atEnd())
    {
        document.write(items.item().Name);
        items.moveNext();
    }
}
```

Figure 6.59: Javascript function using WMI API to enumerate users' names.

When a context metadata record is collected by the sensor, the series of actions depicted in Figure 5.51 occurs: the sensor inserts the sensitive data (i.e. the user context) within the local WMI using the Microsoft™ API, and forwards the non sensitive data (i.e. environment and usage models) to the insert service for indexation into the central context repository. The identification of the user on the central repository is ensured by applying the MD5 algorithm to the first name and last name of the user that are defined within the WMI repository; an adaptive component that needs context information from the central repository to process its adaptation mechanisms has thus to first query the WMI repository in order to generate the MD5 string matching with the target user before sending the appropriate query to the search service of the middleware layer. Moreover, to identify a user that may log into different computers of the same local network, the basic idea is to set up network services such as NIS (Network Information Services) and NFS (Network File System) to save the user profile as a string (using the MOF or xmlCIM formats) on the remote server used to identify users on the network, exactly as bookmarks or general preferences are retrieved from one computer to another.

Storing sensitive data on a local WBEM implementation is a promising solution when it is applied to a specific operating system, but does not ensure the generality of the approach. Even if WBEM became very popular among industrials and software editors, some operating systems do not implement WBEM as default management tool (such operating systems mainly include various Linux distributions). Moreover, even if most of operating systems natively embed a WBEM application, each editor extends the native CIM classes to define their own knowledge, thus requiring sensors to take into account each specific model. Therefore, to fill these lacks, we suggest a web-oriented alternative based on cookies.

6.4.2 Storing Sensitive Data into Cookies

Cookies are widely used by website developers to personalize content provided to users. Basically, a cookie is a small piece of data initially created by a website and then sent to the browser of the user;

whenever the user visits the website, the cookie is encapsulated into the HTTP request so that the website receives the specific information of the user. Web advertisements are most often based on this principle.

Cookies may contain any text information such as the user profile expressed through the MOF or xmlCIM formats (even if xmlCIM represents the best alternative, thanks to the huge number of libraries dedicated to the manipulation of such kind of data). The storage of sensitive data on the local PC thus consists in creating a cookie containing the instances of the classes defined in the user context model.

Here, the identification of the user is achieved by applying the MD5 algorithm to the first name and last name of the user that are mentioned into the AdWLE (in the previous section, this information was given by the local WBEM application): this hash represents the name of the cookie, and identifies the user within the central context repository. Obviously, we suppose here that the user mentions his/her real first and last names in each learning system. When a context metadata record has to be collected, the sensor checks if a cookie matching with the MD5 hash of the user is available. If the cookie is not set it is created by the sensor to store the sensitive data, otherwise the sensor updates the values of the existing xmlCIM instances. Finally, when a user logs into various computers of the same local network (e.g. within a university), his/her personal context is naturally stored on the NFS server alongside other specific data of the browser such as the history or bookmarks.

Compared to the first solution involving a local WBEM application, cookies bring several advantages:

- **Portability.** Cookies are implemented by all browsers and all web scripting languages and do not require an *a priori* framework or specific API.
- **Confidentiality.** As mentioned earlier, cookies' values are encapsulated into the HTTP request; the secured HTTPS protocol can be used to encrypt the data and prevent man in the middle attacks.
- **Scalability.** A given cookie may apply to multiple network domains, thus the same user profile can be shared across various AdWLEs.
- **Ease of use.** Since the user context is automatically sent to the AdWLE whenever the user interacts with the system, sensors or adaptive components do not need to send an additional request to the local host when sensitive data is required.

6.5 Conclusions

This chapter has described an implementation of the distributed architecture presented in Chapter 5, based on open source software. After having tested several kinds of storage system (a pure WBEM im-

plementation, an XML-oriented database and an object-oriented database) to implement the context layer, we have identified the Oracle Object-Relational database as a good candidate offering very interesting response time to complex queries processed over a large amount of data. Oracle O-R combines the advantages of both relational and object-oriented paradigms: the data is modeled as objects and can be easily manipulated through the SQL query language.

The various services of the middleware layer have been developed as SOAP web services using the Java programming language. Concerning the default input/output formats required/returned by these services, we have chosen SQL as query language and SQL2XML as result format for the search service, and XML for expressing the context metadata used by the insert service. SQL and SQL2XML brings flexibility when one comes to query any piece of information stored into repository, whereas a dynamic schema imposes semantic constraints to the collected context metadata. This schema is also flexible, since it suggests various structures for each entity to be observed, according to the context models. We have also exploited the standardized xmlCIM format to represent CIM extensions of the context models as well as CIM instances describing definitions of new indicators.

Among the various components that may be integrated into the AdWLE layer, we have focused on sensors since we consider these plugins as cornerstones of the whole framework: they represent providers that populate the context repository from which all other components are based on. Despite this, we have created an adaptive component for a specific application (ARIADNE Finder) detailed in Chapter 8. No model designer modules have been implemented, instead we have developed a standalone GUI application to allow the definition of new indicators together with the identification and selection of context entities for which inferred data has to be calculated (see Chapter 7). To date, we have also implemented an indicator listener in Moodle able to receive a given indicator (the proportion of actions detailed in Chapter 7), the subscription being performed manually.

Two approaches have been implemented to store sensitive data on the device of the user. While native WBEM implementations integrated into most nowadays systems (WMI in our experimentation) do not ensure the generality of the approach, the storage of sensitive data on the local device using cookies brings several advantages: portability, confidentiality, scalability and ease of use. In addition, the fact that we have placed source code on the user device lets us the possibility to explore information related to hardware and software components available to the user (from the WBEM implementation natively installed), and thus to enhance the user profile and offer new perspectives of adaptation.

Chapter 7 - Sources of Context Data

Contents

In this chapter we present the learning systems which have been observed through our framework to date, together with the matching learning resources and activities. We also expose various indicators that have been calculated on the basis of the collected context.

7.1 Moodle from IUT A Paul Sabatier

The first learning environment interacting with our context framework is the platform Moodle (Modular Object-Oriented Dynamic Learning Environment) deployed at IUT A Paul Sabatier. This platform is fully integrated into the traditional training process, and enables teachers to submit learning resources, assignments and various complementary activities to traditional ones.

7.1.1 The Moodle Environment Model

On the basis of the elements of the environment model exposed in Chapter 4, we derived various classes to take into account the context specific to Moodle. The resulting environment context illustrated in Figure 7.60 comprises several components of the learning management system, from courses to forum discussion and assignments:

- **TEL_LearningManagementSystem** depicts a LMS and allows to represent Moodle as well as any other learning management system. The specific properties of a LMS, in addition to the attributes inherited from the super-class, are related to the main e-learning standards (SCORM and IMS-LD) that could be taken into account by such a system.

- **TEL_Courseware** represents the main component of a LMS, as it is the highest granularity level resource handled by a LMS. The *Category* property specifies the educational category (such as economics or computer science) to which the courseware belongs to. A courseware is associated to a LMS through the *TEL_IsDeployedBy* composition relation.

- **TEL_LearningObject** depicts the learning units used to build the content of a course, and comprises eight new properties referring to the SCORM, IMS-LD, IMS-Qti and IMS-Content packaging standards. A learning object is part of one or several courses, this composition being modeled by the class *TEL_IsPartOf*.

- **TEL_Forum** represents the discussion forums associated with a course where students perform collaborative interactions on one or more subjects related to the topics of a course. It is associated with a courseware through the composition class *TEL_ForumInCourse*.

- **TEL_Discussion** describes the discussion threads within a forum, and is associated to this last through the composition class *TEL_DiscussionInForum*.

- **TEL_Message** models the messages which compose a discussion thread, and is linked to the matching class through the relation *TEL_MessageInDiscussion*. This class contains one property: the content of the message. As the message may include email addresses, real names and living / working places it is subject to privacy concerns. These concerns are not considered at the moment. Additional information such as the sender and the receivers can be retrieved from other classes of the model as presented in the next section.

- **TEL_Assignment** refers to tasks assigned to students by their teachers. Common assignments include problems to be solved, presentations to be built, or other skills to be practiced. The nature of the assignment is specified by the property *Type*, whereas its content by the *Content* property. The class *TEL_AssignmentInCourse* translates the integration of an assignment into a course.

- **TEL_Submission** represents submissions of students in response to the assignments given by teachers. A submission is linked to an assignment through the composition class *TEL_Submission-InAssignment*.

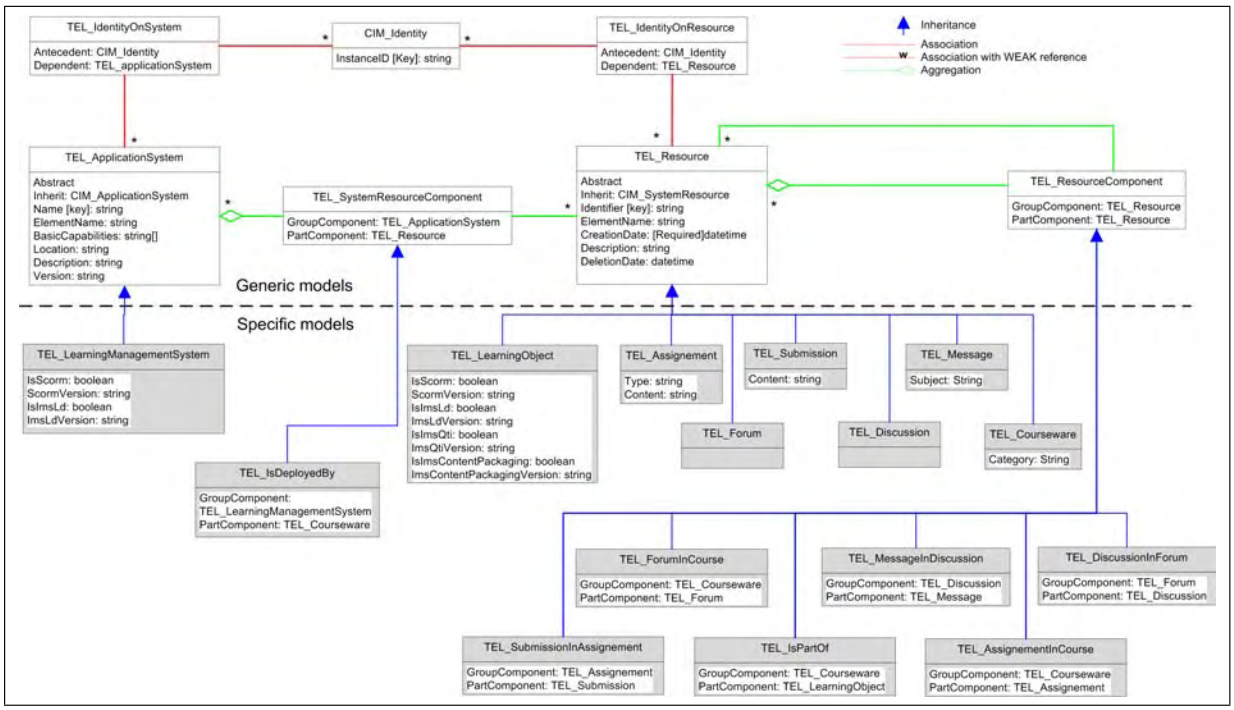


Figure 7.60: The environment context specific to Moodle.

7.1.2 The Usage Model

The usage context illustrated on Figure 7.61 specifies the activities associated to the elements specified in the previous section, on the basis of the usage model introduced in Chapter 4. As already stated, the first level of abstraction is designed to specify activities performed over any system and resource (through respectively *TEL_SystemActivity* and *TEL_ResourceActivity*), whereas the second level of abstraction is related to a given system or resource; thus, one root activity class is associated to each entity introduced in the previous section. Finally, the last level of abstraction describes the concrete activities that can be performed over each element of the environment context. The resulting usage context thus comprises the following activities:

- Two activities are defined for LMS: *has logged in* and *has logged out* respectively indicate that a user has successfully logged in/out the system.

- Five activities are related to courses: *consultation*, *subscription*, *creation*, *update* and *deletion*.

- The activities dedicated to learning objects comprise *consultation*, *download*, *rating* given by students, *integration* within a course, and *removal* from a course.
- The activities related to an assignment include: *download* of the assignment by a student in order to solve it, *consultation*, *update* and *deletion*.
- Two types of activities are defined for a submission: *creation* (by a student as response to the assignment) and *consultation* (by a teacher).
- The activities specific to a forum include *creation*, *consultation* and *update*.
- The discussion thread activities comprise *creation*, *update* and *deletion*.
- The following activities characterize a message: *posting* a message into a discussion thread, *updating* a message, *consulting* a message, and *deleting* a message from the thread. The sender of a message matches with the person who performed the *posting* activity, while the receivers are the students enrolled into the course. Even if the consultation activity is observed, it is very difficult to detect if the message has been effectively read. The number of consultations of a message can be retrieved by exploring the *TEL_DependencyResourceActivity* class.

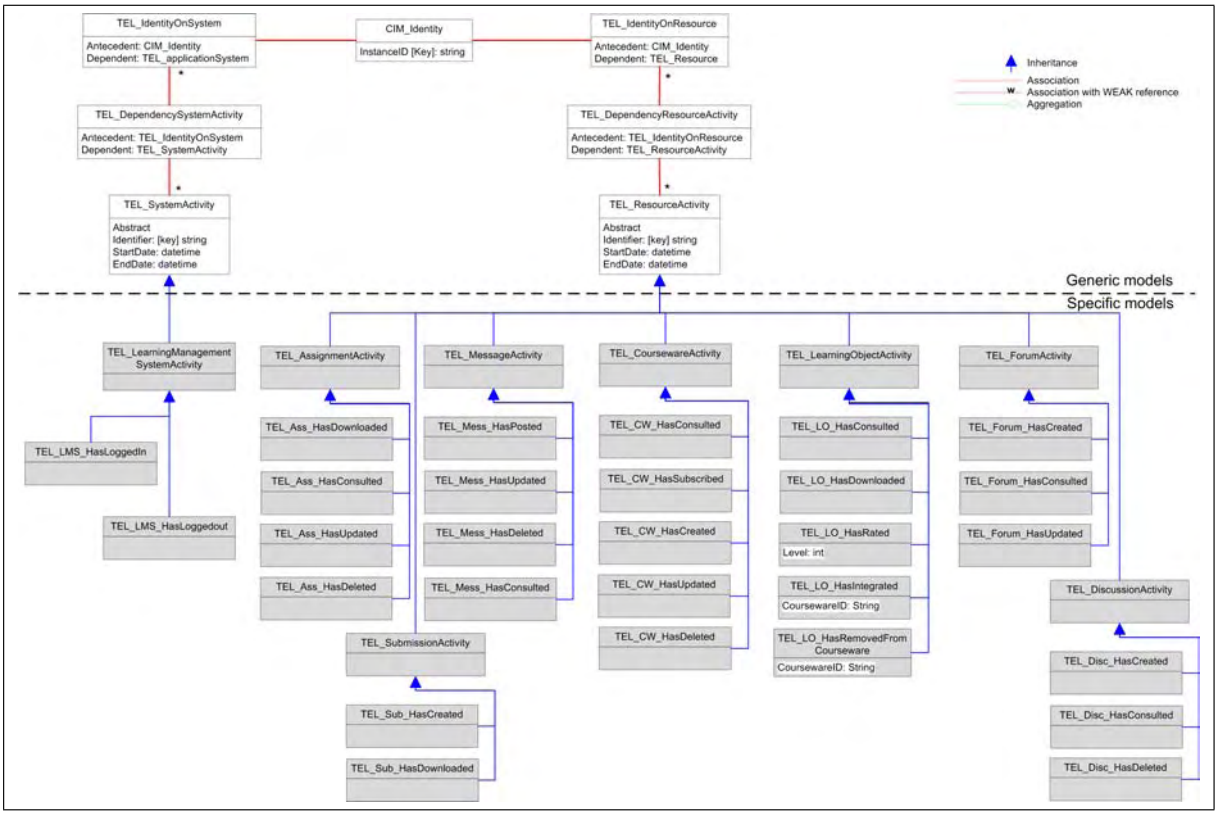


Figure 7.61: The usage model specific to Moodle.

Starting from the specific environment and usage contexts described above, Table 7.19 exposes the number of activities that have been collected [BVB11b]; the user context has not been extended, since the model illustrated by Figure 4.34 comprises the required elements to describe a Moodle user. Supervision of the platform began in early January 2011, and these statistics correspond to a period of one year. The relatively large amount of occurrences is explained by the large number of users of the platform (about 6000), and the use of this tool by students and teachers of all sixteen departments of IUT.

Resources or applications	Activities	Context metadata content	Number of activities
Platform (1) Moodle-IUT	Login, Logout	User profile (U) + properties of the platform (P)	1042850
Courses (1988)	Subscription, creation, consultation, modification, deletion	U + P + properties of the course (C)	1602667
Learning objects (18289)	Consultation, download, rating, integration, deletion, indexation	U + P + C + properties of the learning object (L)	1137021
Assignments (2602)	Download, consultations, update, deletion	U + P + C + properties of the assignment (A)	334431
Forum (7322)	Creation, consultation, update	U + P + C + properties of the forum (F)	8268
Discussion threads (5500)	Creation, consultation, deletion	U + P + C + F + properties of the discussion thread (D)	77151
Submissions (1860)	Creation and download	U + P + C + D + properties of the submission (S)	126620
Messages within discussion threads (1275)	Post, update, deletion, consultations	U + P + C + F + D + properties of the message (M)	84464

Table 7.19: Observed activities and collected metadata.

7.1.3 Indicators Calculation

Based on this large amount of attention metadata, we defined elementary, arithmetic and complex indicators including various statistical information such as the total number of consultations, downloads or ratings for a given learning object or courseware. Among these indicators, we defined the Proportion of Actions Indicator (PAI). This indicator identifies the role taken by each participant during a collaborative learning process: it calculates the amount of actions produced by each member in a group who acts on a set of resources. Since the PAI relies on a variable number of intermediary indicators (e.g. the number of activities performed by each subject on each learning object of a course), it cannot be calculated using elementary nor arithmetic indicator; we modeled the PAI as a complex indicator, as shown on Figure 7.62 that characterizes the proportion of actions of subject S1 over the learning objects contained by the course CW1. The indicator is calculated for an association between a user and a resource; thus the indicator definition applies on the class *TEL_IdentityOnResource*.

The PAI for a subject S1 is calculated through the following formula:

$$PAI(S1) = \frac{\sum_j S_1 A_j}{\sum_{i,j} S_i A_j} \tag{7.1}$$

where S1Aj is the total number of actions performed by the subject S1 on the resource Aj, and SiAj is the number of actions performed by the subject Si on the resource Aj. The PAI(S1) is thus the ratio between the whole set of activities performed by S1 on each resource of a given course and the sum of activities performed by each subject enrolled into the course on the set of resources of this course. Here is the algorithm for calculating this indicator:

1: $SA \leftarrow 0, A \leftarrow 0$
2: $indicator\_value \leftarrow 0$
3: $loIDs \leftarrow select\ PartComponent.Identifier\ from\ TEL\_IsPartOf\ where\ GroupComponent.Identifier = CW1$
4: **for** each loID in loIDs **do**
5: $sAct \leftarrow select\ count(*)\ from\ TEL\_DependencyResourceActivity\ where\ Antecedent.Dependent.Identifier$
6: $= loID\ and\ Antecedent.Antecedent.InstanceID = S1$
7: $SA \leftarrow SA + sAct$
8: $Act \leftarrow select\ count(*)\ from\ TEL\_DependencyResourceActivity\ where\ Antecedent.Dependent.Identifier$
9: $= loID$
10: $A \leftarrow A + Act$
11: **end for**
12: $indicator\_value \leftarrow (SA * 100)/A$

The proportion of actions algorithm

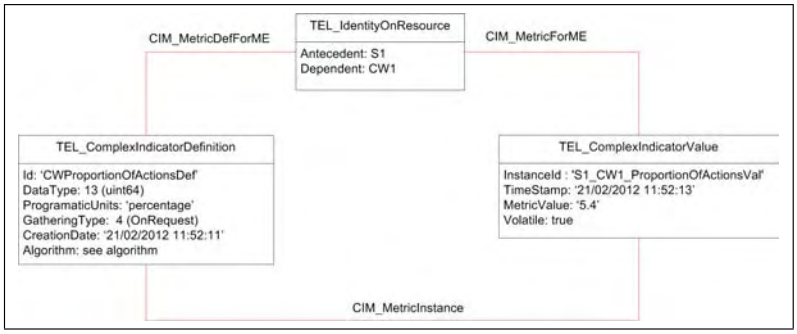


Figure 7.62: Proportion of actions indicator definition.

This indicator definition can be further reused either with the same purpose but applied to other users and/or courses (only the creation of a new *CIM_MetricDefForME* instance is required), or as intermediary indicator for calculating more complex indicators.

Figure 7.63 shows the definition of this indicator using our indicator management client application. The left side exposes a form matching with the attributes of the class *TEL_ComplexIndicatorDefinition*, including the algorithm to calculate the indicator value. The right side of the figure comprises a drop down list exposing the whole set of classes of our models, and allows to select the class the indicator applies to. When the class *TEL_IdentityOnResource* is selected, another drop down list appears so that the designer is able to select the course for which he/she wants to calculate the indicator. Once

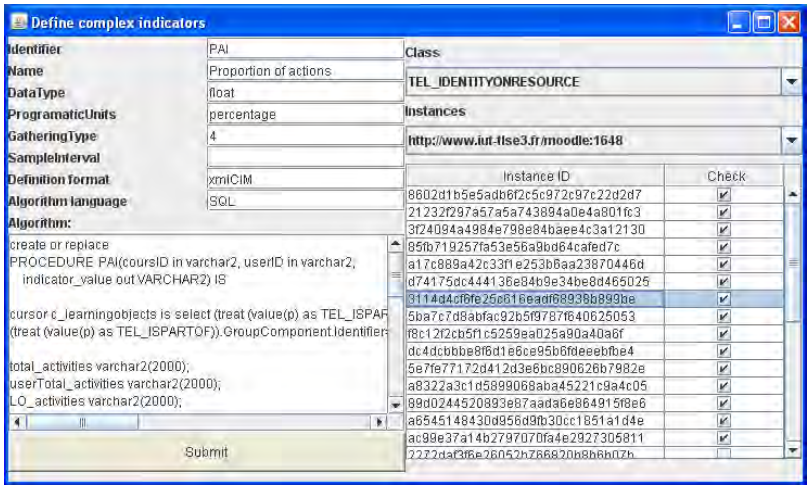


Figure 7.63: Definition and calculation of the PAI.

a course is selected, a table exposing all users that interacted with that course is displayed and the designer can select the students on which the PAI will be calculated. Finally, the algorithm is compiled as a stored procedure, the calculation is processed for the selected students, and the matching instances are created (i.e. *TEL_ComplexIndicatorValue*, *CIM_MetricForMe* and *CIM_MetricInstance*). Figure 7.64 exposes the calculated values.

Let us note that the application we developed is stand-alone; it is not integrated within a learning application. Therefore, the identities of the users are displayed as stored into the context repository. However, if the indicator is calculated from inside a learning application, the teacher can see the names of the students instead of their encoded values.

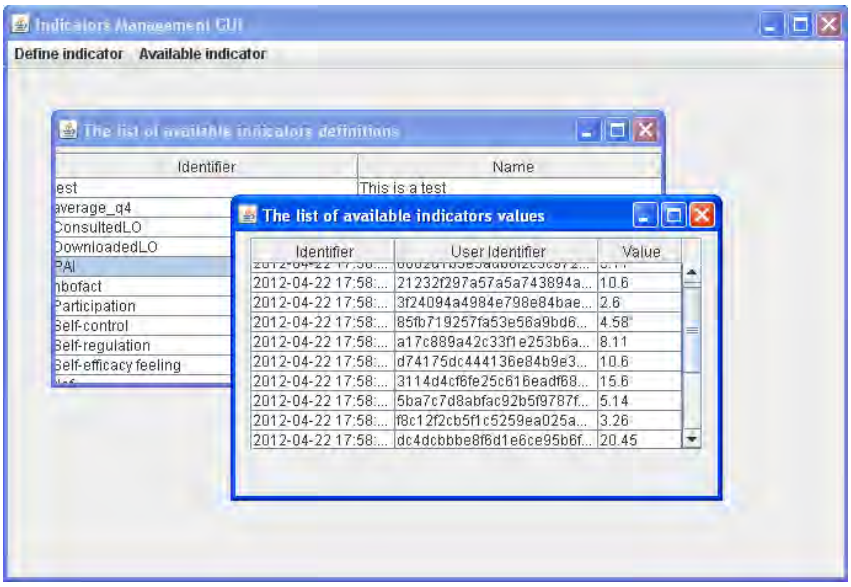


Figure 7.64: Listing the values of the indicator.

Based on the PAI, mirroring and guiding tools can be designed. Mirroring tools can be created by providing the value of the indicator for both teachers and students; teachers can identify the students with very low PAI and further guide them to improve the value of the indicator, whereas students can visualize both their own PAI and the ones of their colleagues and thus become motivated to increase their PAI. Guiding tools can be created as well by comparing the value of the PAI with predefined values to perform actions accordingly (e.g. to provide advices, hints, helps, etc.).

7.2 The CONTINT Project

Our second experimentation was conducted under the CONTINT project. The CONTINT project, funded by the French National Research Agency (ANR), aims at implementing a quality initiative related to a global re-engineering process for the acquisition and development of skills and competencies. The consortium comprises two transversal research laboratories (The Institute of Computer Science of Toulouse - IRIT - UMR 5505, and Cognition, Language, Languages, Ergonomics, Laboratory Work and Cognition - CLLE-LTC - UMR 5263) and two organizations providing the experimental ground (the company TIRESIAS-SCF, and The Professional Insertion Help Desk - BAIP - Paul Sabatier University, Toulouse).

Our responsabilities in this project consist in collecting the context generated by learners engaged in a certification process, but also by instructional designers and producers of educational activities and resources. The second phase consists in transforming the collected context in order to calculate relevant indicators and to help various actors in achieving their tasks:

- The learner identifies his strengths and weaknesses and is continuously developing an action plan that allows him/her to gain quick, committed and effective skills that he/she is missing;
- The tutor has the elements needed to support learners in defining their action plans, and to improve the guidance of activities;
- The instructional designer enhances the proposed activities, and adds new ones becoming relevant due to the enrichment from the user experience;
- Similarly, resource producers improve their educational offer.

Some experiments were conducted as a series of quizzes developed as interactive web pages and integrated within a Moodle server; our task was to gather the interactions of the subjects with these quizzes using our framework.

The data to collect about such a quiz illustrated in Figure 7.65 includes the questions and propositions of each question, together with a set of activities such as start or end a quiz, check or uncheck a proposition, go to the next or previous question, etc.

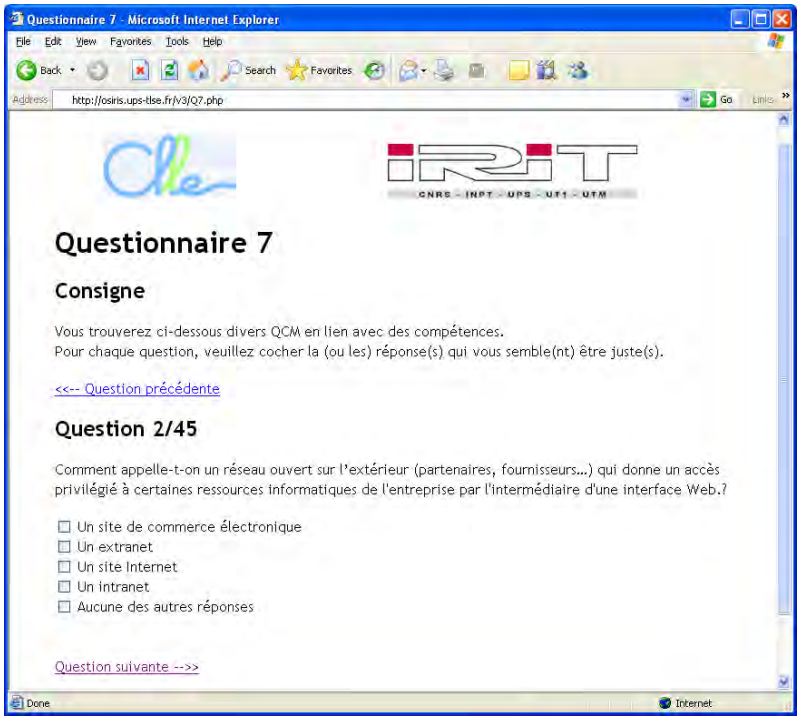


Figure 7.65: A question of one of the CONTINT project quizzes.

7.2.1 The Environment Model

Figure 7.66 illustrates the environment model that describes the context of a quiz in terms of resources, systems and relationships. In addition to the classes already detailed before, it comprises:

- **TEL_Quiz** represents a quiz and is attached to a course through the association *TEL_Courseware-QuizComponent*.
- **TEL_Question** describes a question of a quiz, and is linked to the later through the class *TEL_Quiz-QuestionComponent*.
- **TEL_Proposition** is intended to describe a proposition and is linked to a question through the class *TEL_QuestionPropositionComponent*.

7.2.2 The Usage Model

Concerning the usage model, activities that can be performed over each class introduced above are defined. These appear in gray color on Figure 7.67 and comprise:

- Six activities related to a quiz: *has consented* (to denote the (dis)agreement of a user to complete a quiz), *has started* (to denote the beginning of a quiz), *has finished* (to denote the completion of

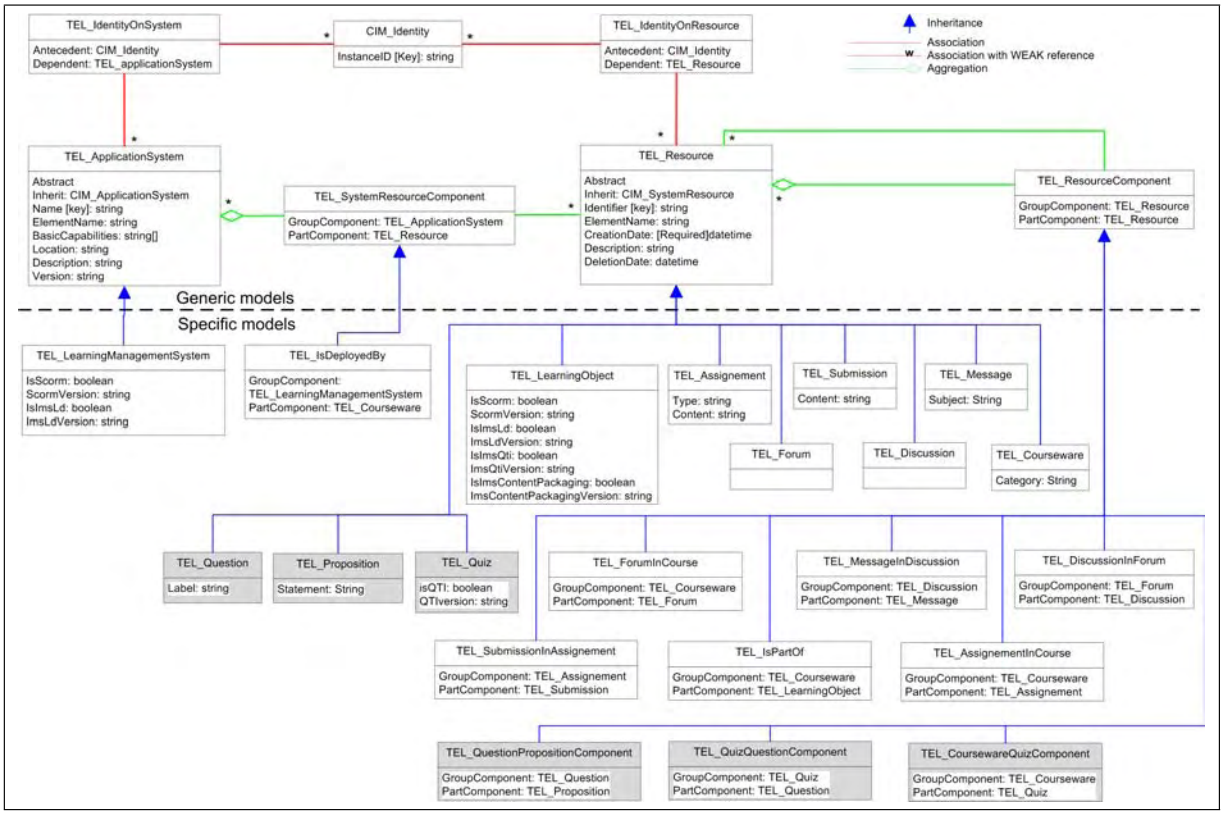


Figure 7.66: The environment model specific to a quiz.

a quiz; if a result is computed it is stored within the *Result* property), *next/previous question* (to denote the navigation to the next/previous question of a quiz), and *jump to question* (to denote the navigation to any question of a quiz);

- Two types of activities are linked to a question: *consultation* and *mouse freeze* (the user didn't move the mouse during more than 30 seconds);
- The *checked* and *unchecked* activities are specified for a proposition.

7.2.3 Collected Data

The experiments of this project address the IT and Internet Certification Level 2 - Engineering Profession (C2I2MI) with three types of participants:

- Academic students of the Master M1-M2 MIAGE;
- Professional employees of the TIRESIAS-EFC company benefitting from IRT (Individual Right to Training);

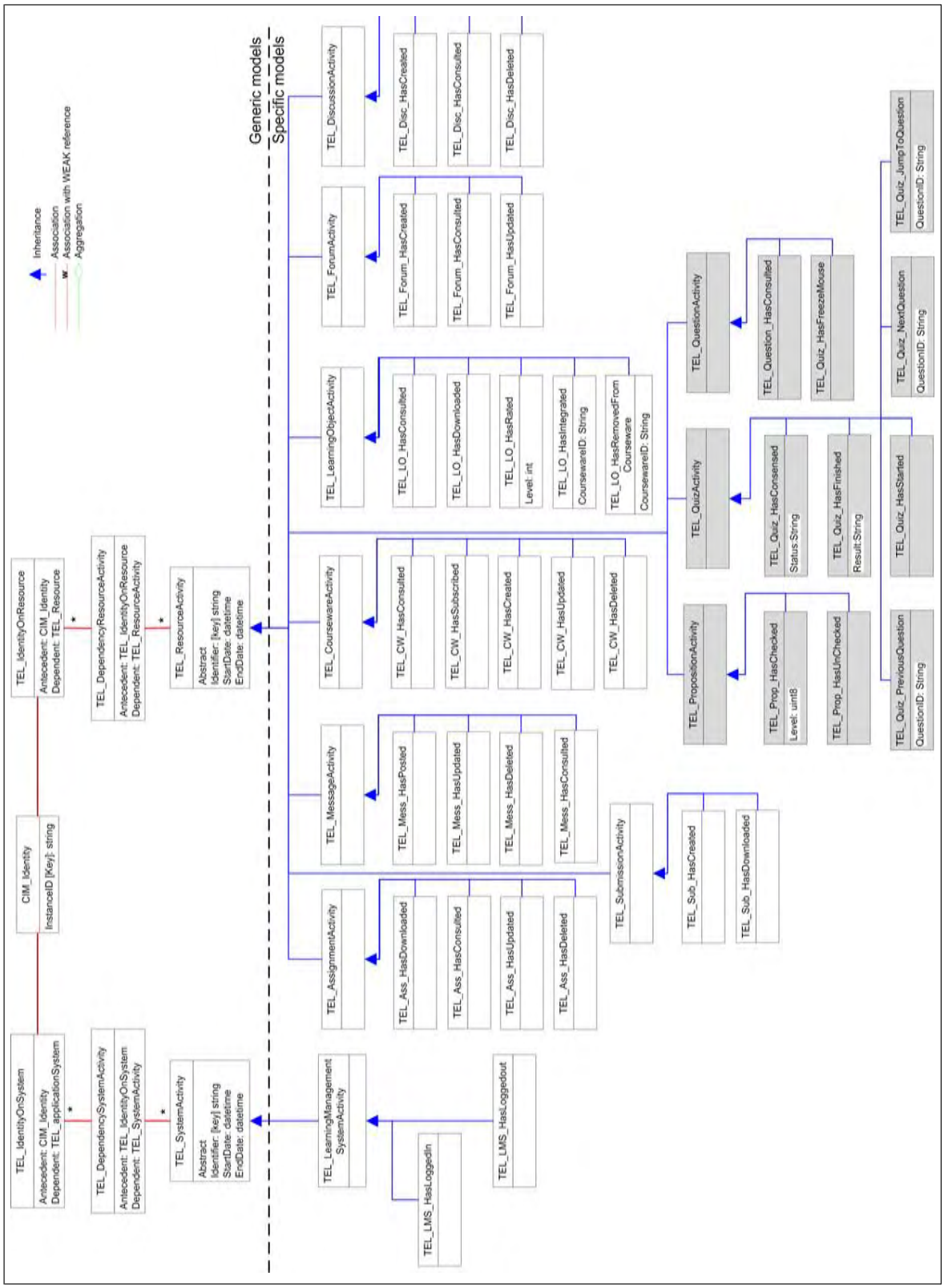


Figure 7.67: The usage model specific to a quiz.

- Hybrid students of the Master M1-M2 MIAGE Alternate Training (MIAGE AT). The alternate training implies the enrollment into an academic cursus, and in the same time the employment into a company.

Exp. Date	No. Users	Group	No. Quizzes
17/02/2011	18	MIAGE AT 2010-2012	11
09/03/2011	7	TIRESIAS 2010-2012 1/2	11
27/04/2011	7	TIRESIAS 2010-2012 1/2	11
16/12/2011	15	MIAGE CT 2011-2013	19
19/01/2012	35	MIAGE AT 2011-2013	19
26/01/2012	15	MIAGE CT 2011-2013	4
09/02/2012	10	MIAGE AT 2011-2013	6
23/02/2012	11	TIRESIAS 2011-2013	19

Table 7.20: The experimentations of the CONTINT project.

The experimentations of the project are spread along 4 years (2011 - 2014), and Table 7.20 shows these which took place to date and observed using our framework. The collected context metadata during the eight experimentation sessions is summarized in Table 7.21. The 19 quizzes are part of a course hosted by a dedicated Moodle platform and comprise 227 questions including around 400 propositions.

Resources or applications	Activities	Context metadata content	Number of activities
Platform (1) Moodle-Osiris	Login, Logout	User profile (U) + properties of the platform (P)	261
Quiz (19)	Consentement, Start, Finish, Next question, Previous question, Jump to question,	U + P + properties of the course (C) + properties of the quiz (Z)	12682
Question (227)	View, Freeze mouse	U + P + C + Z + properties of the question (Q)	28144
Proposition (395)	Checked, Unchecked	U + P + C + Z + Q + properties of the proposition	28787

Table 7.21: Observed activities and collected metadata.

7.2.4 Indicators Calculation

Based on the collected context metadata, various indicators have been defined for the CONTINT project. As an example, quiz number 4 is based on the Motivated Strategies for Learning Questionnaire (MSLQ), a self-report instrument designed to assess college students' motivational orientations, but also the way they use different learning strategies [PDG90].

Table 7.22 shows the indicators that have been defined for this quiz, together with their description and the questions on which they are calculated (the whole set of questions of this quiz appear in Ap-

pendix D). Each indicator is calculated as the average of students' answers for the associated questions; an average greater than 57% validates the ability described by the indicator for a given student.

Indicator name	Description	Questions
Task Value	Students' opinion about the importance and usefulness of the learning task. High task value should lead to more involvement.	1, 4, 10, 13
Extrinsic Goal Orientation (rewards)	Represents the degree of involvement of a student to achieve a goal. An extrinsically motivated student performs because of rewards that are external to the activity itself.	2, 5, 8, 11, 14
Extrinsic Goal Orientation (failure)	The same as previous, except that an extrinsically motivated student performs because of punishments that are external to the activity itself.	3, 6, 9, 12
Self-efficacy feeling	The judgement of a student about his ability to accomplish a task as well as his confidence in his skills to perform that task.	15, 17, 19, 22 25, 31, 34, 37
Self-control	Control of learning refers to students' beliefs about their effort to learn. If students believe that their efforts to study make a difference in their learning, they should be more likely to study strategically and effectively.	16, 24, 27, 33
Help seeking	The capacity of a student to identify someone (peers or teacher) able to provide him/her with some assistance when he/she doesn't know something.	18, 28, 30, 32
Self-regulation	Students' ability to continuously adjust their cognitive activities: change the way of learning, read again a topic, etc. Regulating activities are assumed to improve performance by assisting learners in checking and correcting their behavior.	20, 21, 26, 36
Planning	Planning activities help to activate relevant aspects or prior knowledge that make organization and understanding of the material easier.	23, 29, 35

Table 7.22: Indicators calculated from a quiz.

All definitions of these indicators have been introduced into the context repository through the stand-alone application presented before, and specified as complex indicators: Figure 7.68 illustrates the modeling of the Self-Efficacy Feeling (SEF). They will be further analysed and exploited by researchers of the CLLE-LTC specialized in cognition and metacognition for learning strategies.

7.3 ARIADNE Finder

The third application that we observed is the ARIADNE Finder illustrated on Figure 7.69 that provides a federated search over distributed learning objects repositories. The Google™-like interface contains a text field where the user inserts one or more keywords, and a list of learning objects related to keywords is displayed to the user. The web-based interface also provides the opportunity to filter the list of learning objects according to their provider, language or format (see left-hand side of Figure 7.69), and to search in additional repositories (see right-hand side of Figure 7.69).

The observation of this application may help, for example, to evaluate the quality of a learning object and of a repository.

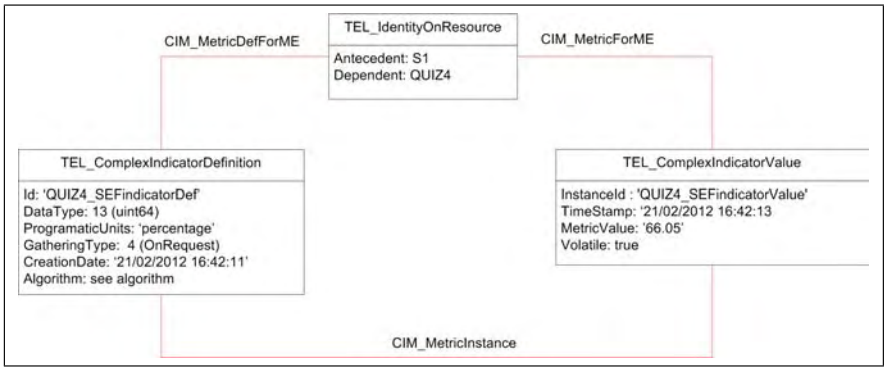


Figure 7.68: The SEF indicator definition.

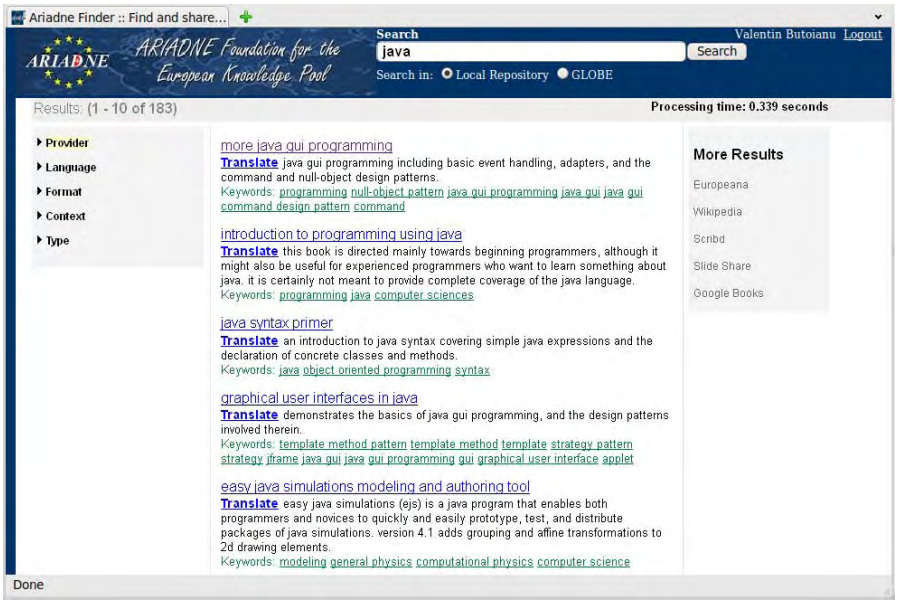


Figure 7.69: A screenshot of the ARIADNE Finder.

7.3.1 The Environment Model

The entities of the environment context involved in the ARIADNE finder are the learning objects and the learning objects repositories. Since LOs are already modeled, only the definitions of a LOR and the relation describing the storage of a LO into a LOR were required to take into account the ARIADNE Finder context; these classes are highlighted in Figure 7.70.

From now, a learning object has two usage contexts: it can be observed as part of a course deployed within a learning management system, and/or stored into a LOR. Our approach thus allows sensor designers and end-users applications developers to choose the appropriate usage context(s) which fits their needs best.

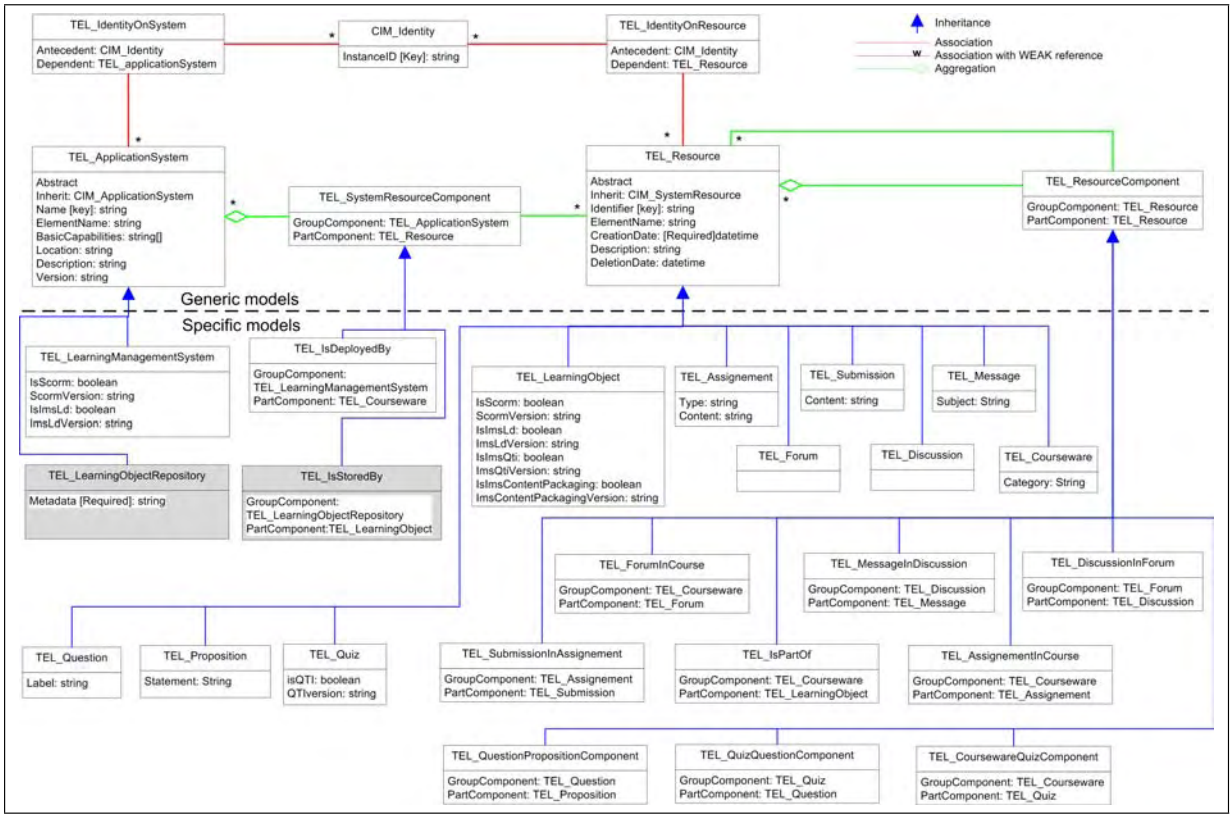


Figure 7.70: The ARIADNE Finder environment model.

7.3.2 The Usage Model

The latest version of the Finder allows users to search for learning objects only. However, a previous version provided users, beside the search feature, with the possibility to index learning objects into a repository as well. Since we integrated a sensor into both versions, we introduced two activities specific to search and indexation into the usage context (see Figure 7.71). Let us note that other activities specific to LOs (i.e. consultation, download, etc.) were modeled in the usage context of Moodle.

7.3.3 Collected Data and Calculated Indicators

We collected data from a local instantiation of the ARIADNE Finder. Because of a low visibility of the server, the collected data is not significant (hundreds of activities only). Several simple statistical indicators such as the total number of LOs indexed into a LOR or the number of LOs deleted from a LOR have been calculated.

7.4 Conclusions

In this chapter we have highlighted two features of our framework: (1) the extensible capacity of the models to represent context specific to new applications, and (2) the capacity to define and calculate TEL indicators on the basis of the collected context.

Starting from the generic context models introduced in Chapter 4, we have defined new classes to represent the context of three heterogeneous applications and demonstrated how our approach is able to federate specific contexts. The unified view of users' context among various applications is meant to enhance the adaptation process since it represents the users attention over a multitude of applications (instead of focusing on a single AdWLE).

Then, on the basis of the collected context, we have designed a series of indicators through a dedicated application, from simple ones such as the total number of activities performed over a learning resource to complex ones such as proportion of actions. The indicators are used by AdWLEs designers, teachers and students. The AdWLEs designers are using the indicators to create adaptive services on their basis, teachers use them to evaluate both students and learning scenarios. Students use the indicators as a mean to be aware about their actions and to position themselves among their colleagues. At the moment, the indicators designers are computer specialists, but on the future, thanks to some GUI tools that we plan to develop, teachers and students will be able to design their own indicators too. Moreover, let us mention that the location of a given resource can be retrieved from the environment context. Therefore, it is possible to use the context repository as a Learning Object reFeratory (LOF): not only resources specific to an AdWLE can be exploited, but also resources from a federation of AdWLE, as demonstrated in the next chapter. Indeed, it shows how a personalized recommendation system can be built on the basis of the context data stored within our framework.

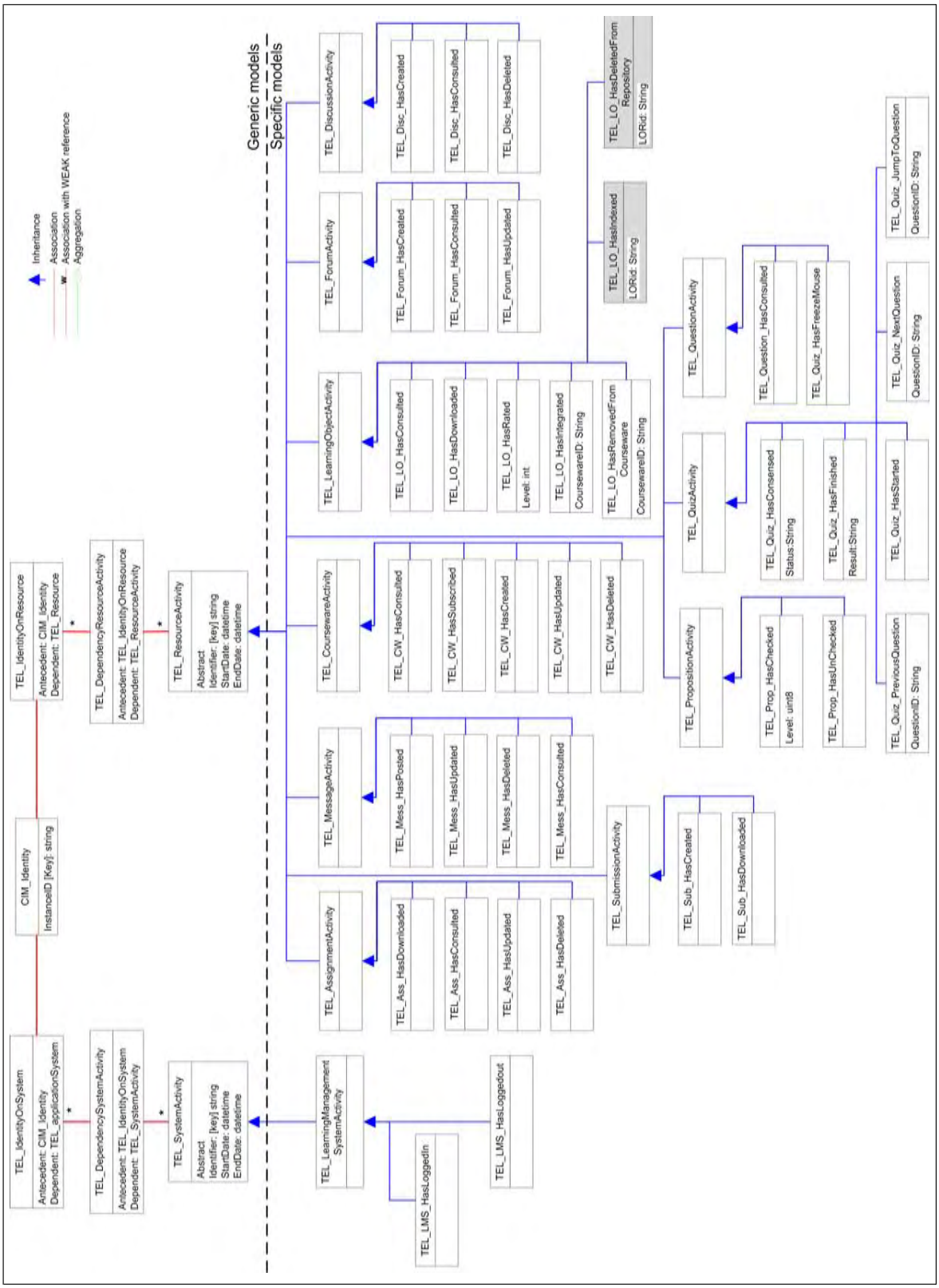


Figure 7.71: The usage model specific to ARIADNE Finder.

Chapter 8 - Applications Built upon our Framework

Contents

This chapter presents two applications which have been built on top of our context framework. The first one is a personalized system for recommending documents to students according to both their current activity and the semantic annotations associated to the resources [BBB+10]. The second application is a visualization application (i.e. mirroring tool) based on multi-touch technology to facilitate the navigation among collected context metadata.

8.1 A Personalized Recommendation Framework based on Context and Document Annotation

The recommendation technique presented here considers annotations associated to each document visited by a user in order to identify ontology concepts that express the current goal of the user; based

on these concepts, relevant resources are recommended to the user. The original aspect of this recommendation approach consists in combining a user activity tracking system with the exploitation of semantic annotations associated to learning resources. Indeed, the user profiles of common recommender systems focus on user navigation activity considered in terms of items, pages, documents, etc.

In order to meet the requirements of such a recommendation system, the environment context has to be extended to take into account semantic annotations and topics of interests, whereas an additional service has to be designed into the middleware layer to implement the recommendation algorithm; the user context, as it appears on Figure 4.34, does not need any extension. Indeed, the *Knowledge* property of the class *TEL_LearnerKnowledge* represents the current users' concepts of interest regarding some given ontologies specified in the *OntologyRef* property. The semantic web community defined XML, RDF or OWL versions of the ACM, ODP (Open Directory Project), or ECDL (European Computer Driving Licence) taxonomies in order to express user profiles in terms of learning competencies in the computer science domain. Since we are interested in modeling the user goals, we chose the ACM ontology to express the learning objectives a user is reaching. Thus, the first element of *OntologyRef* refers to this taxonomy, while the first element of *Knowledge* represents an array of real numbers where each value indicates the interest of the user regarding each concept specified in the ACM taxonomy (the size of the array matches with the number of ACM concepts); depending on how relevant is a concept for the current user goals, the matching value varies from 0 to 1. This modeling results in a matrix Upxm, where U[k,j] represents the weight of the concept j in the profile of the user k, k=1,p; j=1,m.

8.1.1 Extending the Environment Context

E-learning systems use ontologies to exclusively annotate materials, or apply annotations regarding e-learning standards [AKD06]; various relation types were adopted in order to refine the ontology-based annotations of learning objects [EHLS06]. Existing techniques for document annotation according to a domain model are inspired by some classic (web) information retrieval techniques that exploit hypertext features such as hyperlinks and HTML tags. Moreover, the term-based document indexation was possible due to the latent semantic indexing technique [SKKR02], or to some knowledge representation models and methods that are typical to artificial intelligence domain (such as neural networks, semantic networks, Bayesian networks) [MSM07].

Thus, our recommender system takes advantage of a previously reported document indexation technique able to generate the ontology-based annotations of a given textual document. The mentioned technique combines the matrix singular decomposition method with WordNet-based keywords processing. Given a document collection and an ontology, this technique provides, as result, a matrix Dnxm where each row corresponds to a document i, each column to a concept j, and D[i,j] represents the weight of the concept j for the document i (i=1,n; j=1,m).

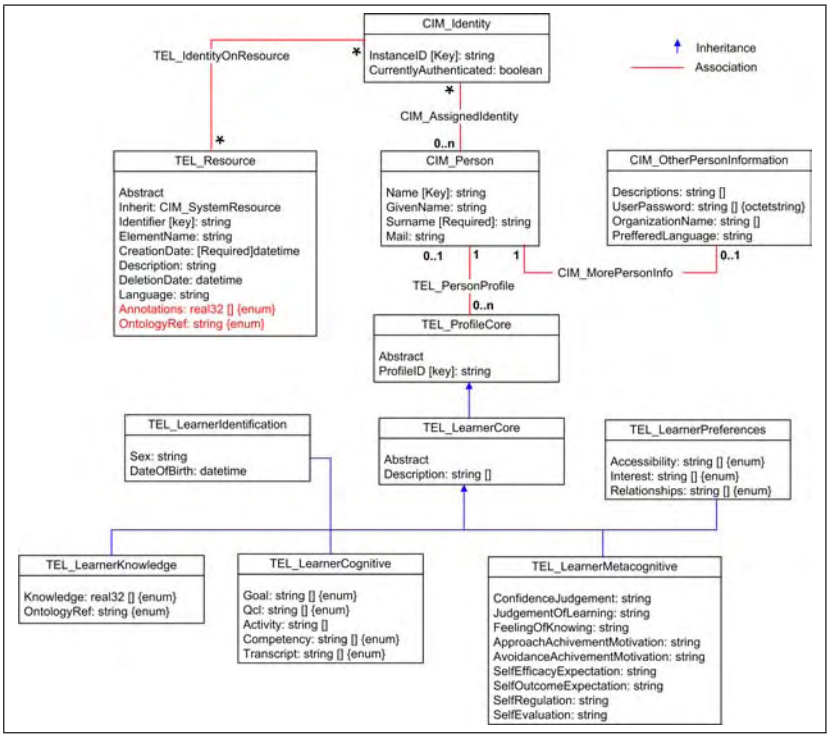


Figure 8.72: An extract of the context model: resources and users.

To take into account the semantic annotations of learning resources into our context models, the *Annotations* and *OntologyRef* properties have been added to the class *TEL_Resource* (see Figure 8.72); they have the same meaning than the properties describing the knowledge of a user, but apply to a document (the matrix D representing annotations is thus defined as an enumeration of arrays, each of them describing the relevance of the concepts appearing in the matching ontology).

8.1.2 The Recommendation Algorithm

The particularity of our recommendation approach consists in supervising the user conceptual navigation inside the ontology instead of his/her site navigation: for each visited document, the associated annotations define the user current goals (as ontology concepts). Therefore, on the basis of the two matrix used to represent the users and collections of documents (respectively Upxm and Dnxm), the following recommendation algorithm is applied:

1. At the beginning of the working session, all values of the *Knowledge* property of the learner k are null: U[k,j] = 0, $1 \leq j \leq m$;

2. When the user access a document k0, the weights of the ontology concepts (stored into the *Annotation* property of the document model) are incremented to the user profile: U[k,j] += D[k0,j] , $1 \leq j \leq m$;

3. A document filtering is accomplished according to the user profile, and more precisely according to the concept c[j0] characterized by the highest weight in the user profile. Furthermore, only the documents k' having $D[k', j0] \neq 0$ are considered. Thus, user localization is accomplished with respect to his/her conceptual navigation inside the ontology.

4. Among the considered documents, the nearest neighbors of the document D[k0] visited by the user are identified using the similarity function [SM86]. The cosine similarity between the document D[k0] and a document D[i] is :

$$sim\left(D_{k0}, D_i\right) = \frac{D_{k0} \bullet D_i}{|D_{k0}||D_i|} \tag{8.2}$$

5. As a reaction to these recommendations, the user choose to consult another document. If this document is one of the recommended documents, the algorithm re-iterates the second step.

6. If the document is not a recommended document, the user may have changed the focus of his/her goal. Thus, in the development of the user profile, we maintain a certain degree of importance for the previous focused concept, alongside those granted to the new one(s). Step 3 is re-applied to identify the new concept c[j1] that expresses the user goal, and then the similarity between c[j0] and c[j1] is computed according to the formula provided by [LBM03] (and proved to be the best similarity measure between two concepts belonging to a semantic network):

$$sim\left(i_1, i_2\right) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}, if\ i_1 \neq i_2 \\ 1, otherwise \end{cases} \tag{8.3}$$

   where l = the length or the shortest path between c[j1] and c[j2],

   h = the number of layers (height) in the ontology between i1 and i2,

   $\alpha = 0.2$, $\beta = 0.6$ are proved as being the optimal values.

7. This similarity function is applied to the user profile in order to reduce and increase the matching weights. At this point, the second step is reiterated.

To instrument this algorithm, a new service has been built as part of the context framework, alongside the services of the middleware layer. The service is thus independent from the target learning systems and can serve various applications, as demonstrated in the next section.

8.1.3 Use case: Recommending Learning Objects

To exploit the recommendation service, we reused the ARIADNE Finder presented in the previous chapter by integrating an adaptation component able to communicate with the recommendation ser-

vice. Thus, the ARIADNE Finder comprises two components: a sensor responsible for collecting context metadata (see Chapter 7) and an adaptive component responsible for providing end-users with personalized content [BCVB11] [BVB11d].

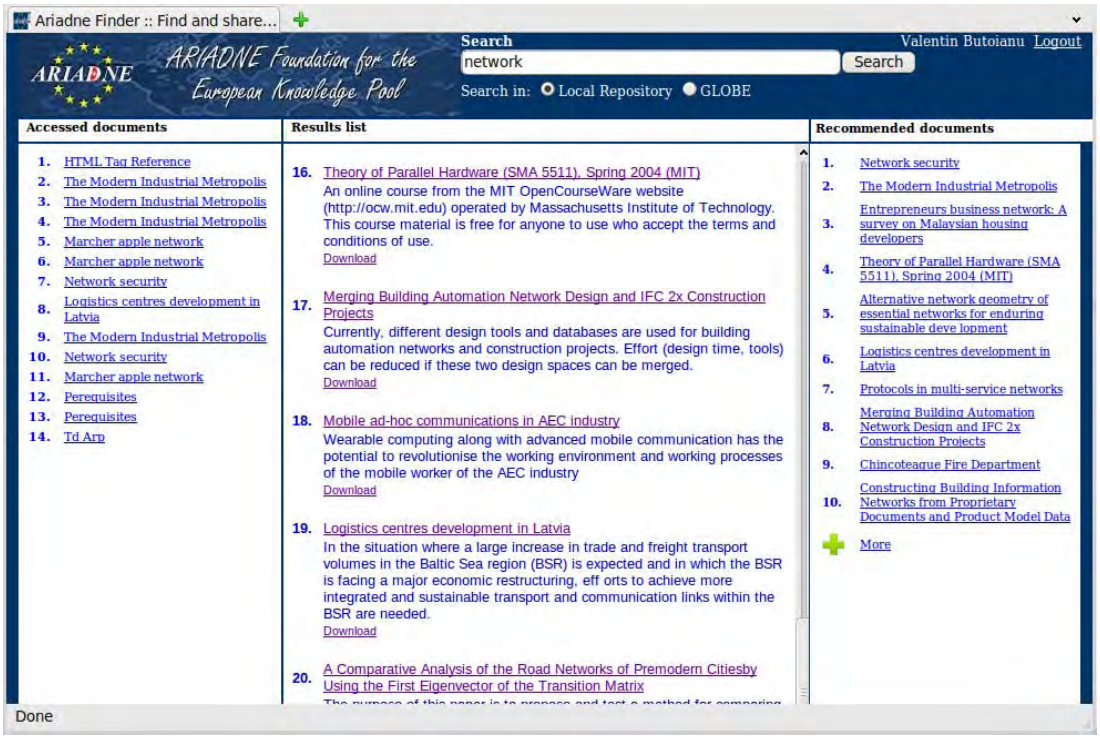


Figure 8.73: Synchronous recommendation through the ARIADNE Finder.

Synchronous recommendation is performed when the user is connected to the ARIADNE Finder. When a user consults a document, the following series of actions is performed:

1. The sensor collects the context metadata describing the activity of the user, sends the non-sensitive data to the insert service and forwards the sensitive data (the user profile) to the users' browser through a cookie;

2. The adaptive component retrieves the current user profile from the local device and sends a request to the recommendation service;

3. The recommendation service then invokes the search service to retrieve the annotations of the learning resources stored into the context repository;

4. It applies the recommendation algorithm to match the user profile with the returned resources to select the most relevant ones;

5. It finally forwards the documents to the adaptive component of the ARIADNE Finder.

The user interface of the ARIADNE Finder is thus divided into three main parts, as illustrated on Figure 8.73: the frame on the left contains the documents accessed by the user, the main frame relates the GLOBE documents matching with the search criteria, and the frame on the right presents the list of recommended documents; the content of this last frame evolves each time whatever action is performed by the user.

8.2 Multi-touch Tracking Visualization System - MultiTravle

To be able to browse the context repository, we designed a visualization application based on multi-touch technologies. These technologies consist of a touch pad or a transparent touch surface applied on a screen, making the device able to capture different points of contact and thus determine the movements on the touch pad.

MutliTravle stands for Multi-touch TRAcking VisuaLization systEm. The concept of MultiTravle is simple: to navigate through any type of data stored into the context repository as simply as possible. To achieve such a result with a dynamic data model (the context models can be extended to meet specific needs), the navigation model is recursive and allows navigating from components to sub-components to display various information.

The application consists in a single window composed of two drop-down menus and a main navigation plan, and supports three types of tactile motion: the “hit” (the simple touch of a finger, the equivalent of releasing the mouse button), the “double hit” (two consecutive “hits”) and the “scaling” (expansion of a graphic object with two fingers). The navigation is performed according to three types of plans: the home plan, the plan of resources and the plan of users. From an entity on a plan, all components of the same type are displayed on the new plan, and one of these components can be detailed to view its sub-components, etc. The backward return is made available by one of two drop-down menus across a cache system that allows instant access to a previous plan. When viewing a large number of entities of the same type (in this case, paginated for better readability), it is possible to add search filters to easily refine the search.

8.2.1 The Visualization Model

As mentioned above, the visualization model illustrated on Figure 8.74 distinguishes three types of entities: the systems, the resources enclosed within these systems, and the users which perform activities on these systems or resources. Let us remember that one system can be composed of sub-systems, and one resource of sub-resources.

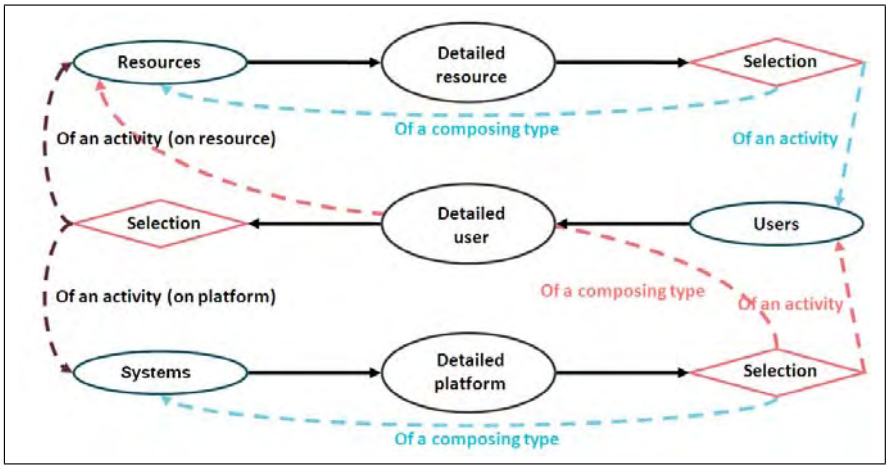


Figure 8.74: The visualization model.

8.2.1.1 The Plan of Systems

When the application is launched, the plan of systems appears and displays a world map illustrated on Figure 8.75 to geographically locate the systems. This is possible thanks to the *Location* property of the class *TEL_ApplicationSystem* combined with a geolocation service to retrieve the matching GPS coordinates. By double hitting one of the spots on the map, the details of the systems are displayed; they include the attributes of the class *TEL_ApplicationSystem*, the total number of users registered into that platform, together with all types of activities that have been performed on the system and the matching number of occurrences. When the platform is zoomed enough, the spot will transform itself into a pie chart where slices represent the types of resources the current platform is hosting; the size of each slice depends on the number of instances of the given type of resource.

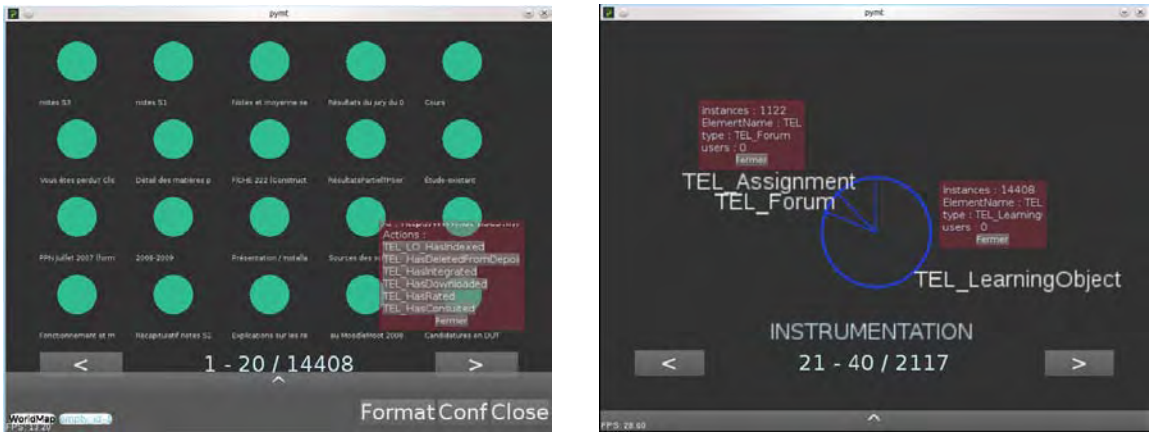
From the plan of systems, the user can further navigate to the plan of resources by selecting one of the slices, or to the plan of users by selecting a given activity.

8.2.1.2 The Plan of Resources

This plan displays the instances of a given type of resource. On Figure 8.76a, the pagination shows only 20 resources on a page thanks to the configuration mechanism of the search service. A double hit on a resource displays details such as the attributes of the class *TEL_Resource* or the different types of activities performed over the resource.

When the plan is zoomed enough, the pie chart applies again, where slices represent the types of sub-resources that compose the current resource (see Figure 8.76b). Thus, the recursive navigation into the resource plan makes it possible to navigate through all granularity levels of a resource.

Figure 8.75: Zoom on the world map.



(a) View of resources of the same type

(b) Detailed view of a selected resource

Figure 8.76: The plan of resources.

8.2.1.3 The Plan of Users

The plan of users illustrated on Figure 8.77 is accessible from both other plans, and displays all users who performed the selected activity over the system or resource. A double hit over a user icon shows the different types of activities the selected user performed over the system or resource.

By selecting an activity type, the user will be redirected to one of the two other plans, depending on the type of the selected activity. Finally, a recursive navigation into three plans is offered to end-users.

Figure 8.77: The users who interacted with a selected resource.

8.2.2 Architecture

The main component of MultiTravle is the RequestManager, since it is responsible for processing all requests. First, it loads at startup the XSD model corresponding to the default context format (see Chapter 6) so that the SchemaManager can automatically build the hierarchical tree representing the various entities defined within our models. Then, the RequestManager performs the authentication (i.e. the obtention of a session identifier) and query configuration (e.g. set query language, set results format, set max query results, set results set size) phases. Furthermore, when information from the context repository has to be shown, the RequestManager sends the adequate queries to the search service, expressed using SQL. Once the response is received according to the result format (xmlCIM for most of the queries), it is analyzed and transformed into data types manipulated by the upper layers of the application.

8.2.3 Implementation

This application has been developed using the Python programming language, designed in the early 1990s by Guido van Rossum. It is a high level interpreted scripting language that can be used sequentially (like C) or as an object-oriented language (like Java). Data manipulation (entirely by reference) coupled with completely transparent typing of data represent the main advantages of this language and make it easy to fastly create advanced applications in a short period of time.

The multi-touch layer is handled by the PyMT (Python Multi-Touch) API, a set of Python scripts dedicated to the management of multi-touch surfaces. In addition, PyMT brings an event handler,

many graphical components (widgets) based on openGL[1], and a manager of animations.

8.3 Conclusions

This chapter showed how our framework for collecting and sharing context can be used by third party applications.

The tool for recommending resources to students according to their current learning goals and activities is based on (1) context metadata collected when a user performs an action on a resource, (2) the user and environment contexts described in terms of semantic annotations referring the ACM taxonomy, and (3) an algorithm able to calculate similarities between these contexts. Compared to user profile generation and maintenance techniques depicted in Chapter 2, we adopted the vector space model which is initially empty and updated as soon as the user starts interacting with the system; data used to update the user profile is provided by the context repository that stores information about users' navigation among a collection of documents (i.e. implicit feedback). The profile learning technique is based on the context of the learning resources the user interacts with, since a resource is represented as a vector of weighed concepts calculated from the occurrence of the concepts within a document and a collection of documents (TF-IDF). The adaptation model is using a hybrid approach: content-based filtering between the user profile and the collection of documents (to retrieve documents that might be of interest to the user), and collaborative-based filtering (to find neighbor documents).

MultiTravle, a context visualisation application based on multi-touch technologies, interacts with the middleware layer in order to browse the context repository and to display its containing data. The visualization is performed on three plans that are interconnected to provide a recursive navigation through infinite zoom from a plan to another.

The possibilities to build other applications or services on top of our framework remain opened. Within our research team, one of the colleagues started a PhD thesis to build an ITS on the basis of the context framework.

[1] Open Graphical Library - http://www.opengl.org

CONCLUSIONS

Chapter 9 - Conclusions and Future Works

Contents

The works presented in this manuscript were meant to address some actual challenges of existing AdWLEs [VMO+12], and particularly focused on the share and reuse of context metadata at a large scale. To reach these goals, our contributions relate on three main proposals: (1) a unifying model able to federate data coming from heterogeneous AdWLEs, (2) a distributed architecture based on standard interfaces and loose-coupling components, and (3) some organisational principles and guidelines that ensure confidentiality of sensitive data describing actors of the framework. This chapter gives a brief overview of these contributions and exposes, for each of them, some future works.

9.1 A Unifying Model to Federate Heterogeneous Context Metadata

The object-oriented model we have elaborated to federate context resulting from the interactions of users with heterogeneous applications is based on the Common Information Model standard, and comprises three main interconnected sub-models: the user context depicts users in terms of learning style, knowledge or preferences, the environment context details information about adaptive systems but also learning resources hosted by these applications, and the usage context describes actions of users over systems and resources. Our model is characterised by a fixed structure of high abstraction level which provides the same backbone for all observed applications, but also allows the definition of detailed information specific to a given application through the extension of the backbone; this modeling results in a good generality-usability compromise. Moreover, we have integrated a model dedicated to the definition of indicators, or inferred data, based on the entities of the three sub-models. We have

made a distinction between the indicator's definition, its value, and the entity it applies to: a single indicator definition may apply to any entity of the context models, and may have several values for a given entity. This conception allows to precisely define the meaning of inferred data, to promote their reuse over various entities, and to facilitate the design of complex indicators.

Our modeling approach has been validated through the federation of context metadata provided by three different applications: the learning management system Moodle, a web-based quiz tool, and the ARIADNE Finder dedicated to search and indexation of learning objects into several learning object repositories. On the basis of the context metadata collected from these tools, we have computed various indicators from simple ones such as the total number of activities performed over a given resource, to complex inferred data such as the proportion of actions or the self-efficacy feeling.

The CIM meta-model suggests several qualifiers to add semantics to the observed entities. One of them is the *mapping string*, which aims at matching a given property of the model with the path of the same property in other context metadata specifications. This mechanism offers the opportunity of ensuring various mappings at the model level without any dedicated component. On the other hand, some initiatives such as dataTEL[1] try to make contextual information publicly available. The dataTEL Challenge invited research groups to submit existing datasets from TEL applications to be used as input for adaptive web-based learning environments. To date, six datasets have been collected from Mendeley[2], APOSDLE-DS[3], ReMashed[4], Organic.Edunet[5], MACE[6] and Travel well[7]. Since each of these datasets provides context metadata according to their own proprietary format, we plan to study how these formats can be federated through our model, and then to apply mapping strings to provide the dataTEL initiative with multiple ways of browsing context metadata.

Another perspective relates on aggregated activities, since the aggregation of activities can conduct to the highest activity of the user (the task he/she is performing) and provide users with task-based adaptation techniques and algorithms. As an example, an activity such as "editing a file" is composed of an ordered sequence of four smaller activities: opening the file, modifying the file, saving the file and closing the file. Our model does not currently offer any mechanism to aggregate different activities into higher level activities. However, the CIM policy model [DMT03] allows the definition of basic rules structured as follows: *IF condition(s) THEN action(s)* that could lead to the aggregation of activities. Indeed, *Conditions* are related to events occurring inside the model (e.g. creation/modification/deletion of a class/instance/property), whereas *actions* refer to operations to perform on the model (e.g. create/modify/delete a class/instance/property). In the above example, the four events matching with the creation of the low-level activities would result in the creation of the high-level activity.

[1] http://teleurope.eu/pg/groups/9405/datatel/
[2] http://www.mendeley.com/
[3] http://www.aposdle.tugraz.at/
[4] http://remashed.ou.nl/
[5] http://portal.organic-edunet.eu/
[6] http://portal.mace-project.eu/
[7] http://lreforschools.eun.org/web/guest/travel-well

9.2 A Distributed Architecture that Promotes the Share and Reuse of Context

In order to sustain the previous models, we have adopted a distributed architecture which promotes the share and reuse of context metadata, composed of four layers: (1) the context layer contains a repository to store data and a set of components dedicated to the management of indicators, (2) the middleware layer offers an easy access to the context repository, (3) the AdWLE layer comprises the systems and tools from which context information is collected, and (4) the user layer represents the device used to access the adaptive web-based learning environments. After some performance evaluation experimentations, the context layer has been implemented through the Oracle Object-Relational database that brings compatibility with our object-oriented modeling of context. The middleware layer is based on a SOA which comes with its well-known advantages: interoperability (clients may use their preferred programming languages to interact with the context repository), aggregation (complex services may stand on other simpler services), reusability (the same service can be reused to build other services) and fault tolerance (services may be deployed anywhere at any time in case of failure). Four core services currently compose this layer, where our attention focused on standard interfaces and formats:

- The search service to easily browse the context repository is not specific to a given query language or result format, since it is based on the CEN Simple Query Interface. SQL is the current query language, but other languages can be taken into account as well (results are returned according to the SQL2XML or SQL2xmlCIM formats);
- The insert service to index context metadata implements the SPI interface and suggests two result formats (the add-hoc SQL2XML format, and the standardized SQL2xmlCIM). To ensure semantics constraints defined within our models, we built a dynamically up-to-date XML schema according to our specifications;
- The model management service offers the possibility of extending the core models. Extensions must be expressed using xmlCIM, the native DMTF representation of CIM with XML;
- The indicator service stands on the publisher-subscriber paradigm to notify interested applications of the calculation of indicators as soon as a new value is calculated within the context repository.

To interact with these services, we have extended several open source learning systems with various components; some of them are responsible for collecting context information from the system (they are called sensors) while others are responsible for handling the adaptation process. The well-known LMS Moodle and a specific quiz tool embed a sensor only, while the ARIADNE Finder integrates both

a sensor and an adaptive component to recommend personalized content to users, according to their current objectives; these developments validate the distributed approach, but also show how higher level services can extend the features of the middleware layer, since an additional service reuses the search service to retrieve relevant content. Finally, two proposals have been set up to implement the user layer. They are detailed in the next section.

Even if the context layer can be easily modified to expose a distributed database that handle load balancing to face with scalability issues that may appear, we have to consider modern databases such as document based databases, also called Not Only SQL, that have proven their efficiency when a system comes to deal with a huge amount of data. Currently SOAP web services are only available to request the middleware layer. In order to make our framework lighter and faster in terms of response-time but also to increase access to the context repository, we plan to implement services using the REST architecture. Also, to facilitate integration of sensors within existing application and promote adoption of our framework, we plan to offer libraries intended to developers. But most importantly, we have to focus on end-users to offer teachers, and even learners, with an intuitive way of defining new indicators that are of their interest. This feature includes complex mechanisms and tools: a graphical user interface exposing the existing indicators and offering the possibility of defining new ones, a pseudocode language that should be understood by the middleware layer, and a visualisation interface reflecting the value of the indicators at the right place at the right time. This set of features would greatly improve the way to reuse context metadata, and requires the exploration of modern techniques such as machine learning that brings accurate results [Sch08]. Indeed, it has been successfully applied to infer various data: disengagement detection [CW09], gaming the system detection [CHB09], hidden attitude toward learning [AW05], motivation [HN08] [CW07], performance prediction [KPP04] or learning styles detection [ÖA09].

9.3 Ensuring Users' Privacy

In order to ensure privacy of users within our framework, we have chosen to store sensitive data (i.e. the user context) on their local device (the user layer), and other context metadata (i.e. the environment and usage contexts) on the central repository (the context layer). We have implemented two different solutions for recording users profiles: a WBEM implementation natively integrated within the Windows operating system, and cookies. If the first approach brings a full compliance with the DMTF standards, it can hardly be deployed within any device such as smartphones or tablets that are becoming more and more natural to visit web-based learning environments. The storage of sensitive data on the local device using cookies consists in representing the user context using the standardized xmlCIM format; this solution ensures portability, since this mechanism to store user specific data is widely implemented within any browser and web scripting language, but also offers confidentiality, scalability of

ease of use. In addition to this organisational principle, we also introduced requirements that must be taken into account by the various components of the architecture to respect the legislation related to privacy of users.

Collaborative filtering is one of the most well-known adaptation techniques [Bur07]: based on similarities between users (same interests, preferences, etc.), systems generate recommendations. Lots of web sites implement this feature to provide users with additional links about items that can be of their interest. Even if our framework allows to recommend resources based on the navigation of users within the observed systems, it prevents recommendations based on their profiles, since this information resides on the client machine; for a given user, it would be relevant to provide him with learning paths that have been successful for other users characterized, as an example, by the same pool of competencies. In order to remove this limitation, some studies must be led to identify the components of the user model (e.g. knowledge) that can be shared on the central repository without compromising users identity. Moreover, at the first view, the environment and usage contexts do not contain confidential data about users. The environment context includes the systems and resources that have been in the focus of a user at a certain moment, whereas the usage context describes activities that have been performed on resources and systems by the user. At a glance, such information can be stored without risks on the tracking repository to be shared and reused by other AdWLE client applications. Instead, at a closer look, a question arises: may the linkage of context reveal users' identity? Indeed, a single context metadata record has no significant meaning, but the linkage of an important number of records may have. The more applications and resources are observed, the more detailed information are known about a user, and the higher the chances to identify a user are. Thus we have to perform analyses over context provided by multiple sources to detect if the linkage of context may reveal users' identity.

APPENDICES

Appendix A - A MOF File

Here is the TEL_LearningObject class described in the MOF language.

```
// ==================================================================
// TEL_LearningObject
// ==================================================================
[Description ("Learning object")]
class TEL_LearningObject: TEL_Resource {

[Description("Is the learning object a Scorm module?")]
boolean IsScorm;

[Description("Scorm Version")]
string ScormVersion;

[Description("Is the learning object an ImsLd module?")]
boolean IsImsLd;

[Description("Ims Ld Version")]
string ImsLdVersion;

[Description("Is the learning object an ImsQti module?")]
boolean IsImsQti;

[Description("Ims Qti Version")]
string ImsQtiVersion;

[Description("Is the learning object an ImsContent Packaging module?")]
boolean IsImsContentPackaging;
```

```
[Description("Ims Content Packaging Version")]
string ImsContentPackagingVersion;

};
```

Appendix B - XML Schema Indexation

The XML Schema corresponding to the validation of a Learning Object context metadata.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"><xs:element name="CIM_IDENTITY">
<xs:complexType>
<xs:sequence>
<xs:element name="INSTANCEID" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CURRENTLYAUTHENTICATED" type="xs:boolean"/>
<xs:element name="activityOn">
<xs:complexType>
<xs:choice>
<xs:element name="resource">
<xs:complexType>
<xs:choice>
<xs:element name="TEL_LEARNINGOBJECT">
<xs:complexType>
<xs:sequence>
<xs:element name="IDENTIFIER"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="4000"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

```
<xs:element name="ELEMENTNAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="2000"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CREATIONDATE" type="xs:string" />
<xs:element name="DESCRIPTION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="3000"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DELETIONDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="ISSCORM" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="SCORMVERSION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ISLMSLD" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="IMSLDVERSION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
```

```
</xs:simpleType>
</xs:element>
<xs:element name="ISLMSQTI" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="IMSQTIVERSION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ISLMSCONTENTPACKAGING" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="IMSCONTENTPACKAGINGVERSION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="context">
<xs:complexType>
<xs:choice>
<xs:element name="systemResourceComponent">
<xs:complexType>
<xs:choice>
<xs:element name="TEL_ISSTOREDBY">
<xs:complexType>
<xs:sequence>
<xs:element name="TEL_LEARNINGOBJECTREPOSITORY">
```

```
<xs:complexType>
<xs:sequence>
<xs:element name="NAME"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="500"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ELEMENTNAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="700"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="BASICCAPABILITIES" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="900"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="LOCATION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="600"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DESCRIPTION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="3000"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="VERSION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
```

```
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
<xs:element name="resourceComponent">
<xs:complexType>
<xs:choice>
<xs:element name="TEL_ISPARTOF">
<xs:complexType>
<xs:sequence>
<xs:element name="TEL_COURSEWARE">
<xs:complexType>
<xs:sequence>
<xs:element name="IDENTIFIER"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="4000"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ELEMENTNAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="2000"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CREATIONDATE" type="xs:string" />
<xs:element name="DESCRIPTION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="3000"/>
```

```
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DELETIONDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="CATEGORY" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="context">
<xs:complexType>
<xs:choice>
<xs:element name="systemResourceComponent">
<xs:complexType>
<xs:choice>
<xs:element name="TEL_ISDEPLOYEDBY">
<xs:complexType>
<xs:sequence>
<xs:element name="TEL_LEARNINGMANAGEMENTSYSTEM">
<xs:complexType>
<xs:sequence>
<xs:element name="NAME"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="500"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ELEMENTNAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="700"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="BASICCAPABILITIES" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
```

```
<xs:maxLength value="900"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="LOCATION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="600"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DESCRIPTION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="3000"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ISSCORM" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="SCORMVERSION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ISLMSLD" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="IMSLDVERSION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
```

```
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
<xs:element name="activity">
<xs:complexType>
<xs:sequence>
<xs:element name="device" minOccurs="0">
<xs:complexType>
<xs:choice>
<xs:element name="CIM_KEYBOARD">
<xs:complexType>
```

```
<xs:sequence>
<xs:element name="CAPTION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="2000"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="AVAILABILITY" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CONFIGMANAGERERRORCODE" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CONFIGMANAGERUSERCONFIG" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CREATIONCLASSNAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DESCRIPTION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="500"/>
</xs:restriction>
</xs:simpleType>
```

```
</xs:element>
<xs:element name="DEVICEID"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ERRORCLEARED" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ERRORDESCRIPTION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="INSTALLDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="ISLOCKED" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="LASTERRORCODE" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="NAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
```

```
</xs:simpleType>
</xs:element>
<xs:element name="PNPDEVICEID" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="POWERMANAGEMENTCAPABILITIES" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="600"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="POWERMANAGEMENTSUPPORTED" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STATUS" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STATUSINFO" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="SYSTEMCREATIONCLASSNAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
```

```
</xs:simpleType>
</xs:element>
<xs:element name="SYSTEMNAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="LAYOUT" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="NUMBEROFFUNCTIONKEYS" type="xs:int"  minOccurs="0"/>
<xs:element name="PASSWOARD" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="CIM_POINTINGDEVICE">
<xs:complexType>
<xs:sequence>
<xs:element name="CAPTION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="2000"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="AVAILABILITY" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
```

```
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CONFIGMANAGERERRORCODE" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CONFIGMANAGERUSERCONFIG" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CREATIONCLASSNAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DESCRIPTION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="500"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DEVICEID"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ERRORCLEARED" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
```

```
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ERRORDESCRIPTION" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="INSTALLDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="ISLOCKED" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="LASTERRORCODE" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="NAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="PNPDEVICEID" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="POWERMANAGEMENTCAPABILITIES" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
```

```
<xs:maxLength value="600"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="POWERMANAGEMENTSUPPORTED" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STATUS" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STATUSINFO" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="SYSTEMCREATIONCLASSNAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="SYSTEMNAME" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="HANDEDNESS" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
```

```
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="NUMBEROFBUTTONS" type="xs:int"  minOccurs="0"/>
<xs:element name="POINTINGTYPE" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="RESOLUTION" type="xs:int"  minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
<xs:element name="type">
<xs:complexType>
<xs:choice>
<xs:element name="TEL_LO_HASDOWNLOADED">
<xs:complexType>
<xs:sequence>
<xs:element name="IDENTIFIER"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="500"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STARTDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="ENDDATE" type="xs:string"  minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="TEL_LO_HASINDEXED">
```

```
<xs:complexType>
<xs:sequence>
<xs:element name="IDENTIFIER"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="500"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STARTDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="ENDDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="LORID" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="TEL_LO_HASDELETEDFROMREPOSITORY">
<xs:complexType>
<xs:sequence>
<xs:element name="IDENTIFIER"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="500"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STARTDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="ENDDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="LORID" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

```
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="TEL_LO_HASINTEGRATED">
<xs:complexType>
<xs:sequence>
<xs:element name="IDENTIFIER"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="500"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STARTDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="ENDDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="COURSEWAREID" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="TEL_LO_HASREMOVEDFROMCOURSEWARE">
<xs:complexType>
<xs:sequence>
<xs:element name="IDENTIFIER"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="500"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STARTDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="ENDDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="COURSEWAREID" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
```

```
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="TEL_LO_HASCONSULTED">
<xs:complexType>
<xs:sequence>
<xs:element name="IDENTIFIER"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="500"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STARTDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="ENDDATE" type="xs:string"  minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="TEL_LO_HASRATED">
<xs:complexType>
<xs:sequence>
<xs:element name="IDENTIFIER"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="500"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STARTDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="ENDDATE" type="xs:string"  minOccurs="0"/>
<xs:element name="LEVEL" minOccurs="0"><xs:simpleType>
<xs:restriction base="xs:string">
<xs:maxLength value="100"/>
</xs:restriction>
```

```
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Appendix C - The Result Format

```
<result>
  <row>
    <RNUM>1</RNUM>
    <ANTECEDENT.DEPENDENT>
      <TEL_LEARNINGOBJECT>
        <IDENTIFIER>http://www.iut-tlse3.fr/moodle:1020:1566</IDENTIFIER>
        <ELEMENTNAME>Raccourcis clavier Word 2003</ELEMENTNAME>
        <CREATIONDATE>2011-02-26 14:48:10</CREATIONDATE>
        <DESCRIPTION>Ce cours décrit les raccourcis clavier disponibles
          dans Microsoft Office Word 2003.</DESCRIPTION>
        <DELETIONDATE></DELETIONDATE>
      </TEL_LEARNINGOBJECT>
    </ANTECEDENT.DEPENDENT>
    <DEPENDENT>
      <TEL_LO_HASCONSULTED>
        <IDENTIFIER>2011-05-29 10:36:41.6546</IDENTIFIER>
        <STARTDATE>2011-05-29 10:36:41</STARTDATE>
        <ENDDATE>2011-05-29 10:36:41</ENDDATE>
      </TEL_LO_HASCONSULTED>
    </DEPENDENT>
  </row>
  <row>
    <RNUM>2</RNUM>
    <ANTECEDENT.DEPENDENT>
      <TEL_FORUM>
        <IDENTIFIER>http://www.iut-tlse3.fr/moodle:1016:3677</IDENTIFIER>
        <ELEMENTNAME>Questions sur l'utilisation de l'espace Moodle</ELEMENTNAME>
        <CREATIONDATE>2010-10-29 09:16:19</CREATIONDATE>
```

```
        <DESCRIPTION>Quoi? Ca va pas? Tu comprends pas? T'as besoin d'un manuel?
          Bon si t'as une question sur moodle ou la plateforme c'est ici...</DESCRIPTION>
        <DELETIONDATE></DELETIONDATE>
      </TEL_FORUM>
    </ANTECEDENT.DEPENDENT>
    <DEPENDENT>
      <TEL_FORUM_HASCONSULTED>
        <IDENTIFIER>2011-03-19 14:16:11.3216</IDENTIFIER>
        <STARTDATE>2011-03-19 14:16:11</STARTDATE>
        <ENDDATE>2011-03-19 14:16:11</ENDDATE>
      </TEL_FORUM_HASCONSULTED>
    </DEPENDENT>
  </row>
</result>
```

Appendix D - A Quiz from the CONTINT Project

The questions of the questionnaire Q4, MSLQ.

1. My goal is to learn a lot of new concepts this year.
2. My goal is that other students think I am competent.
3. One of my goals is to not look stupid within this class.
4. One of my goals for this class is to learn as much as possible.
5. One of my goals is to show others that I am competent.
6. One of my goals is to prevent others thinking I'm not smart.
7. One of my goals is to master a lot of new skills.
8. One of my goals is to show others that what is required during the class is easy for me.
9. One of my goals is that the teacher does not think that I know less than other students.
10. One of my goals is to understand in depth the work that I do during this class.
11. One of my goals is to look smarter than other learners.
12. One of my goals is to avoid going for a learner who has difficulties in completing the demanded work during the class.
13. One of my goals is to improve my skills.
14. One of my goals is to look smart compared to the other.
15. I think I will get excellent results in this class.
16. When I study a part of the course, I put questions to myself to guide my reading.
17. I'm sure to acquire the skills taught during the class.
18. Even if I have difficulties assimilating the course content, I'll try to work alone, without any help.
19. I feel I can get very good results to homework and exams of this class.
20. When I do not understand a point of the course, I go back and try to clarify it.

21. If the chapters are difficult to understand, I change my way of studying.
22. Given the difficulty of teaching and my skills, I think I will obtain good results.
23. Before examining in detail a new part of the course, I often run quickly to see how it is organized.
24. I ask questions to make sure I understand the points studied during the class.
25. I am sure I can understand the most difficult points covered in this course.
26. I try to change my way of studying to adapt myself to the demands of the course.
27. When I study, I often realize that I did not pay attention to what I was studying.
28. I ask the teacher for clarification of the concepts I do not understand.
29. When I study, I try in the first step to think about what is important to remember.
30. When I do not understand something, I ask the help of another student.
31. I think I am able to understand the most complex points presented in this course.
32. I am trying to identify the learners to whom I can ask for help if necessary.
33. When I study, I try to see what the concepts that I do not understand well are.
34. I think I will achieve well the course.
35. I set goals to guide my work during each period allocated for study.
36. If the notes I take are not clear enough, I always fix them afterwards.
37. I think I will get excellent results in this class.

Appendix E - French Summary

INTRODUCTION

Chapitre 1 - Introduction générale

Cadre scientifique

Ce manuscrit présente la synthèse de quatre ans de travail effectués dans le cadre d'une thèse. Celle-ci s'est déroulée au sein de l'équipe SIERA[1] du laboratoire IRIT[2] à Toulouse. L'IRIT est une unité Mixte de Recherche (UMR 5505) commune à plusieurs établissements qui sont l'Université Toulouse III Paul Sabatier[3], le CNRS[4], l'INPT[5], et l'Université des Sciences Sociales, Toulouse I Capitole[6].

Les travaux de l'équipe SIERA, dirigée par Abdelmalek Benzekri, visent le contrôle et la maîtrise des infrastructures et services de communication de dernières générations, mais aussi des systèmes et applications complexes dynamiquement agrégés, distribués et trans-organisationnels. Ainsi, ils s'attachent à la définition et à l'évaluation de nouveaux paradigmes de gestion et apportent à la communauté des architectures, des plates-formes, des outils ou encore des contributions à la normalisation. Parmi les différentes thématiques qui font l'objet d'études et de propositions de l'équipe, la distribution et la gestion des environnements de formation en ligne constituent le domaine des travaux présentés dans ce manuscrit. Je présente dans ce document les travaux effectués ainsi que les résultats obtenus.

Contexte de recherche

Son développement rapide, sa popularité et la facilité d'utilisation de ses outils, ont fait du World Wide Web le média le plus important pour la collecte, le partage et la diffusion d'informations [Zai02]. Il n'est

[1] Service IntEgration and netwoRk Administration (Administration de Réseaux et Intégration de Services)
[2] Institut de Recherche en Informatique de Toulouse - http://www.irit.fr
[3] http://www.univ-tlse3.fr
[4] Centre National de la Recherche Scientifique - http://www.cnrs.fr
[5] Institut National Polytechnique de Toulouse - http://www.inpt-toulouse.fr
[6] http://www.ut-capitole.fr

donc pas étonnant que le web soit le choix architectural pour les systèmes de formation à distance avancés. Les systèmes éducatifs permettent aux apprenants d'acquérir des compétences et des connaissances, souvent avec l'aide des enseignants ou des tuteurs. Ils s'appuient sur des outils supports et des ressources pédagogiques, et utilisent le web comme moyen de diffusion de contenus et de partage des connaissances. Ces systèmes sont appelés *Environnements Informatiques pour l'Apprentissage Humain* (EIAH) (ou *Web-based Learning Environments - WLE* en anglais) [ASS+03].

Les avantages bien connus des EIAH classiques reposent sur l'indépendance de la salle de classe ainsi que sur l'indépendance de la plateforme [Bru98]. Une application web peut être utilisée par des milliers d'étudiants partout dans le monde et connectés à Internet, mais l'inconvénient est le manque de personnalisation. En effet, le même contenu pédagogique est proposé à des apprenants issus de milieux et cultures différents.

Pour surmonter les lacunes des systèmes traditionnels, des applications évoluées d'apprentissage capables de s'adapter aux besoins de chaque apprenant ont fait leur apparition ces dernières années *(Adaptive Web-based Learning Environments - AdWLE)* : ces systèmes permettent d'engager l'apprenant dans une stratégie d'enseignement avec des matériaux pédagogiques qui font appel à ses connaissances antérieures, ses besoins, ses objectifs, ses motivations ou ses styles d'apprentissage.

Problématique

Dans une salle de classe traditionnelle, l'enseignant connaît ses élèves en termes de connaissances, motivations, styles d'apprentissage, par observation directe de leur comportement, attention, gestes, mimiques, posture ou intonation. Il peut adapter son scénario pédagogique en rajoutant éventuellement une séquence explicative, un exercice complémentaire, ou laisser un exercice pour la prochaine séance. Bien que cela soit possible dans les salles de classe traditionnelles, cette observation s'avère plus complexe dans les systèmes d'apprentissage en ligne où les différents acteurs sont géographiquement répartis. Dans les environnements d'apprentissage en ligne, l'observation et l'évaluation d'un scénario d'apprentissage sont souvent fondées sur l'analyse d'une grande quantité de données automatiquement collectées pendant la session d'apprentissage. De ce fait, une étape préalable nécessaire aux systèmes adaptatifs pour aider les utilisateurs pendant leur processus d'apprentissage consiste à capturer le contexte dans lequel ils évoluent, dans l'objectif de recueillir une grande quantité de données résultant des interactions des utilisateurs avec les systèmes et ressources pédagogiques. Les AdWLEs réutilisent les données contextuelles collectées pour créer des profils utilisateur et appliquer différents algorithmes d'adaptation pour identifier le contenu pédagogique devant être proposé à un utilisateur spécifique.

Toutefois, compte tenu de l'intérêt suscité par le web social (ou Web 2.0) au cours des dernières années, les utilisateurs se trouvent dans un contexte beaucoup plus ouvert. Ils utilisent des outils et

des services non dédiés à l'apprentissage tels que la messagerie instantanée, les blogs, les réseaux sociaux, etc. La capture des informations de contexte résultant d'interactions entre les utilisateurs et l'ensemble de ces outils rendrait les profils des utilisateurs beaucoup plus détaillés et précis. L'accès à ces données de contexte doit être possible pour différents systèmes adaptatifs. Danc ce cas, les données observées issues de plusieurs systèmes et non d'une unique source permettraient d'améliorer le processus d'adaptation. La question de recherche suivante se pose : *comment les données hétérogènes issues de diverses sources peuvent être représentées de manière unifiée, et comment ces informations peuvent être facilement partagées et réutilisées à grande échelle ?*

Pour répondre à cette question, notre approche repose sur deux propositions principales : un modèle de métadonnées de contexte capable de fédérer les données hétérogènes, et une architecture ouverte support.

Pour présenter nos propositions, ce manuscrit est divisé en trois parties principales : (1) une présentation des AdWLEs suivie d'un état de l'art couvrant les différentes composantes et initiatives ou projets impliqués dans la collecte et le stockage des données de contexte, (2) les solutions que nous proposons pour faciliter le partage et la réutilisation de ces données, et (3) la validation de nos propositions à travers leur déploiement dans le cadre de projets locaux et nationaux. Une conclusion et un ensemble de perspectives clôturent ce document.

PARTIE I - Etat de l'art

La première partie comprend deux chapitres. Le premier se focalise sur l'évolution des EIAH statiques vers les EIAH adaptatifs orientés apprenant pour mettre en évidence la nécessité pour les AdWLEs de recueillir le contexte de l'utilisateur. Le deuxième chapitre expose une analyse de plusieurs approches existantes dédiées à la collecte et au stockage des données de contexte, et met en évidence leurs limites.

Chapitre 2 - Environments Informatiques pour l' Apprentissage Humain

Dans les EIAH classiques, les enseignants / concepteurs utilisent la plateforme d'apprentissage pour créer des scénarios pédagogiques sur la base d'activités ou d'objets pédagogiques stockés dans un espace local ou une base de données externe appelée Learning Object Repository (LOR). Les scénarios sont identiques pour tous les utilisateurs du cours, et leur contenu est statique, inchangé lors de la session d'apprentissage.

En comparaison avec un environnement d'apprentissage traditionnel, les environnements adaptatifs (AdWLEs) surveillent en permanence les activités des utilisateurs dans le but d'adaptation de leur interface et de leur contenu. En plus des deux composants propres aux WLE traditionnels (plateforme et données), un AdWLE contient des composants spécifiques nécessaires à l'adaptation :

- **Modèle de l'apprenant.** Il intègre les caractéristiques de l'apprenant. Deux types d'informations sont représentés ici : (1) les composants indépendants du domaine d'apprentissage comme les informations démographiques, les connaissances antérieures, le style d'apprentissage, les intérêts, les objectifs, et (2) un composant dépendant du domaine d'apprentissage qui représente les niveaux de connaissances de l'apprenant sur les sujets (ou concepts) du domaine proposé à l'étude.

- **Modèle de contenu.** Il comprend (1) un référentiel d'objets pédagogiques en tant que source d'apprentissage, et (2) un modèle de domaine (ou une structure de connaissances, ou une ontologie de domaine), qui est représenté par une hiérarchie de concepts interdépendants, représentant le domaine à enseigner. Les objets pédagogiques sont ensuite associés à des concepts afin de faciliter le processus de (re)ingénierie des scénarios pédagogiques.

- **Modèle de tutorat.** Ce modèle, qui représente également le moteur d'adaptation, intègre les techniques d'adaptation à différents niveaux d'abstraction. Plus précisément, il définit ce qui peut être adapté, quand et comment. Une différence est généralement faite entre macro-adaptation, ou adaptation indépendante du domaine (fondée sur des règles d'adaptation) et micro-adaptation, ou adaptation dépendante du domaine à enseigner (fondée sur des règles d'apprentissage).

La variété des systèmes d'apprentissage adaptatifs est importante. Il est courant de trouver des systèmes qui fournissent les mêmes fonctionnalités, avec des nominations différentes. Trois classes d'applications adaptatives findées sur le web sont décrites dans ce chapitre :

- **Systèmes adaptatifs s'appuyant sur des indicateurs.** Ces systèmes calculent un ensemble d'indicateurs sur la base des activités des utilisateurs observés au sein du système. Ils sont utilisés pour améliorer le scénario d'apprentissage, pour aider les apprenants lorsqu'ils rencontrent des problèmes, et les encourager à participer davantage au processus d'apprentissage en utilisant des outils réflexifs, métacognitifs et de guidage.

- **Systèmes de Tutorat Intelligents (STI).** Un STI s'appuie sur des techniques d'intelligence artificielle et se définit comme un système capable de simuler le comportement d'enseignants pour soutenir les étudiants quand ils sont dans un processus d'acquisition de connaissances [CGCC06]. Habituellement, ces systèmes exploitent le composant du modèle de l'utilisateur dépendant du domaine d'apprentissage et appliquent des règles pour décider si l'apprenant peut aller plus loin dans le processus d'apprentissage ou s'il doit revenir sur certains concepts. Les STI sont des systèmes fermés qui traitent un sous-problème d'un domaine particulier avec un parcours d'apprentissage prédéfini.

- **Systèmes de recommandation personnalisée.** Ces systèmes utilisent les composants du modèle de l'apprenant indépendants du domaine à enseigner pour fournir des contenus personnalisés.

Ces systèmes sont généralement des systèmes ouverts qui utilisent des ressources fournies par divers référentiels et dont le contenu est dynamiquement construit. Ils peuvent utiliser des structures de connaissances pour organiser les sujets traités dans le cours et effectuer la correspondance automatique des ressources ouvertes sur les concepts de la structure. Les nœuds et les liens entre eux sont relativement peu structurés et les apprenants peuvent accéder librement à n'importe quel nœud.

Chapitre 3 - Systèmes dépendants du contexte

Dans ce chapitre, nous avons examiné les systèmes qui modélisent, collectent et stockent le contexte des utilisateurs provenant de leurs interactions avec les différents outils et applications d'apprentissage. De façon plus générale, ces systèmes s'appellent des systèmes dépendants du contexte (de l'anglais, *context-aware systems*). Pour définir notre vision du *contexte*, nous avons fait une distinction entre le contexte de l'utilisateur, le contexte de l'environnement et le contexte d'usage :

- **Le contexte de l'utilisateur** comprend les différentes caractéristiques de l'utilisateur présentées dans le Chapitre 2 (informations démographiques, connaissances antérieurs, style d'apprentissage, intérêts, objectifs, etc.).

- **Le contexte de l'environnement** comprend des informations sur les systèmes ainsi que sur les ressources pédagogiques hébergées par ces systèmes avec lesquels l'apprenant a interagi pendant différentes sessions d'apprentissage.

- **Le contexte d'usage** représente la façon dont l'utilisateur a manipulé les systèmes et ressources observées par le contexte environnemental (consultation, téléchargement, évaluation, etc.). Le contexte d'usage peut contenir également les dispositifs utilisés pour effectuer ces activités.

Les trois dimensions sont reliées entre elles par des relations qui permettent de passer d'une dimension à une autre.

Après avoir donné une définition du contexte, nous allons définir les principes de conception relatifs aux modèles et à l'architecture du système, tout en prenant en compte la protection des informations sensibles concernant la vie privée des utilisateurs dans l'objectif de favoriser le partage et la réutilisation des données contextuelles :

- **Conception des modèles.** Les modèles de contexte doivent être conçus à deux niveaux d'abstraction : le contexte brut directement collecté à partir d'applications observées, et le contexte inféré, calculé sur la base du contexte brut :

- *Un modèle de contexte brut* dédié aux EIAH doit : (1) prendre en compte le profil des utilisateurs, car il représente une condition préalable pour le développement des services adaptatifs, (2) être flexible pour pouvoir représenter le contexte spécifique aux applications et domaines différents, (3) être expressif, pour représenter le contexte à différents niveaux de détails, (4) imposer des contraintes sémantiques afin de réduire le nombre de contextes admissibles, et (5) être extensible pour permettre la définition de nouvelles données contextuelles à la volée (dans le cas où une nouvelle application doit être observée).

- *Un modèle de contexte inféré* doit : (1) être intégré dans le modèle du contexte brut pour faciliter son partage et sa réutilisation, (2) contenir des métadonnées pour décrire les données inférées (comme l'algorithme utilisé pour les calculer), (3) stocker de manière persistante les valeurs calculées de sorte que plusieurs applications puissent les réutiliser, et (4) mettre à jour les valeurs des données inférées.

- **Conception d'architecture.** Les systèmes dépendants du contexte doivent satisfaire aux exigences suivantes : (1) reposer sur une architecture ouverte, afin de permettre l'intégration facile de nouveaux agents de collecte, (2) avoir une architecture distribuée et structurée en couches pour assurer l'hétérogénéité des données et pour faciliter l'extension de fonctionnalités, et (3) permettre le passage à l'échelle compte tenu de la provenance de données de sources diverses.

- **Principes de la vie privée.** Les sept principes de la vie privée établis par la Commission Européenne [Com95] en matière de protection des données à caractère personnel doivent être pris en compte par les systèmes adaptatifs avant leur déploiement au sein des entreprises.

Le Chapitre 3 se termine par une étude des systèmes existants sur la base des critères mentionnés ci-dessus. Les approches étudiées présentent des avantages et des inconvénients, mais aucune d'entre elles n'intègre l'ensemble des critères définis. Nous proposons donc dans la deuxième partie une approche qui prend en compte les avantages des approches existantes et qui comble les manques constatés.

PARTIE II - Partage et réutilisation de données d'observation issues des interactions entre utilisateurs et systèmes d'apprentissage en ligne hétérogènes

Cette partie est également composée de deux chapitres. Le premier présente notre approche pour la modélisation du contexte, le second étant consacré à l'architecture globale support de nos modèles.

Chapitre 4 - La modélisation des EIAH

Dans ce chapitre, nous présentons notre approche pour la modélisation des environnements d'apprentissage fondés sur le web (WLE) et leurs composants. Nos modèles s'appuient sur un modèle commun d'informations (CIM) proposé par le DMTF et dédié à la gestion des applications, systèmes et réseaux. Le méta-modèle CIM et ses schémas représentent une solution uniforme pour modéliser les WLE car il correspond à l'ensemble des exigences de conception de modélisation que nous avons fixées dans le Chapitre 3. Sur la base de CIM, nous avons modélisé le contexte à deux niveaux : le contexte brut, recueilli directement par des capteurs, et le contexte inféré (indicateurs) calculé sur la base du contexte brut.

Modélisation du contexte brut

Le modèle de contexte brut représenté dans la Figure 4.33 comprend trois sous modèles interconnectés :

- **Le modèle de l'utilisateur**, précisément détaillé dans [RVB10], comprend le standard IMS-LIP avec des informations additionelles d'ordre cognitives et métacognitives. En effet, le modèle original décrit par [RVB10] ne contient pas les niveaux de connaissance de l'étudiant sur les concepts du domaine à étudier ; nous l'avons étendu pour prendre en compte ces éléments. Le modèle final comprend tous les éléments du modèle de l'apprenant spécifiques aux AdWLEs que nous avons décrits dans le Chapitre 2.
- **Le modèle de l'environnement** considère des données qui ne sont pas liées aux usagers, mais à l'environnement avec lequel ils interagissent. Ces données concernent les applications ainsi que les ressources manipulées et les relations entre ces applications et les ressources.
- **Le modèle d'usage** relie les deux modèles précédents en décrivant les interactions entre les utilisateurs et les systèmes/ressources pédagogiques. En d'autres termes, ce modèle représente les activités effectuées par un utilisateur sur un système ou une ressource (par exemple, la connexion d'un étudiant à un LMS, le téléchargement d'un objet pédagogique par un apprenant, la modification d'un cours par un enseignant, etc.) et le dispositif utilisé pour effectuer ces activités (par exemple, la souris, le clavier, etc.).

Notre modèle n'est pas limité à une application, puisque de multiples outils et applications peuvent être représentés. Il n'est pas non plus complètement générique et essaie d'atteindre le meilleur compromis généricité - facilité d'utilisation en offrant une vue unifiée des données de contexte hétérogènes. Le caractère extensible de CIM permet à notre modèle d'intégrer de nouvelles données de contexte à la volée lorsque de nouveaux besoins apparaissent.

Bien que ces modèles soient spécifiques aux données de bas niveau, ils sont trop complexes pour être interprétés directement par les enseignants.

Modélisation du contexte inféré

A partir du modèle CIM Metrics, nous avons proposé un modèle générique (Figure 4.40) pour la modélisation des indicateurs. Nous proposons deux types d'indicateurs :

- **Les indicateurs élémentaires** sont calculés directement à partir des métadonnées stockées dans le modèle de contexte brut, sur la base de fonctions mathématiques simples (min, max, moyenne, somme, etc.).
- **Les indicateurs composites** sont calculés sur la base d'indicateurs élémentaires ou d'autres indicateurs composites. Nous avons défini dans un premier temps des indicateurs arithmétiques et complexes :
  - *Indicateur arithmétique :* un tel indicateur est le résultat d'opérations mathématiques complexes appliquées à deux ou plusieurs indicateurs élémentaires et / ou composites. Une formule exprime la façon dont cet indicateur doit être calculé en utilisant les opérateurs traditionnels mathématiques. Ils sont adaptés à un nombre limité d'opérandes, définies à priori.
  - *Indicateur complexe :* ce type d'indicateur ne peut être calculé comme les indicateurs élémentaires ou arithmétiques. Une propriété indique l'algorithme conduisant au calcul de la valeur de l'indicateur, offrant ainsi une liberté totale d'élaborer des indicateurs avancés reposant sur un grand nombre d'indicateurs intermédiaires.

Cette décomposition du contexte inféré permet de réutiliser facilement les indicateurs intermédiaires pour construire des indicateurs similaires, alors que les relations d'agrégation que nous avons introduites apportent de la sémantique. Si la conception d'indicateurs complexes peut s'avérer difficile, une fois qu'ils sont définis, ils peuvent être facilement réutilisés pour s'appliquer sur n'importe lequel des éléments du contexte brut, grâce à la distinction claire entre leur définition et leur(s) valeur(s). D'autre part, cette distinction facilite leur réutilisation par des environnements d'apprentissage adaptatifs : les métadonnées décrivant la définition d'un indicateur rendent facile l'identification précise de la nature et objectif des données inférées.

Chapitre 5 - L'architecture globale

Dans ce chapitre, nous allons présenter notre proposition d'architecture distribuée pour la collecte, le stockage et le partage de données contextuelles provenant d'applications d'apprentissage hétérogènes.

Elle s'inspire de la spécification WBEM (Web-based Enterprise Management) proposée par le DMTF et se compose de quatre couches (voir la Figure 5.50).

La couche des données contextuelles

Cette couche contient un référentiel CIM qui héberge les classes et les instances de nos modèles de contexte, et un gestionnaire qui expose plusieurs interfaces dédiées à la manipulation des données stockées dans le référentiel. De plus, cette couche fournit certains composants nécessaires à la gestion des indicateurs :

- *Le gestionnaire des événements* a un rôle double. Tout d'abord, quand un nouvel indicateur doit être calculé, il avertit le gestionnaire des indicateurs responsable de son calcul. D'autre part, lorsque la valeur d'un indicateur est calculée ou mise à jour, il informe le notificateur des indicateurs de sorte que des actions externes puissent être mises en place.
- *Le gestionnaire des indicateurs* est responsable du calcul des valeurs des indicateurs en fonction de leur définition et assure la création ou la modification des instances correspondantes dans le référentiel de contexte.
- *Le notificateur des indicateurs* a pour objectif de permettre l'exécution d'actions indépendantes du référentiel chaque fois qu'il reçoit une notification du gestionnaire d'événements. Dans notre cas il appelle le service d'indicateurs de la couche intermédiaire qui informera à son tour les applications abonnées à cet indicateur.

La couche intermédiaire

Le but de la couche intermédiaire est de combler l'écart entre les outils de la couche des applications adaptatives et la couche de contexte en offrant un accès facile au référentiel. Cette couche est conçue comme une architecture orientée services (SOA) et comprend 5 services, certains standardisés, et tous indépendants d'un format particulier :

- *Le service de gestion de session* permet aux consommateurs d'établir une session. Elle est requise avant toute communication avec les autres services.
- *Le service de recherche* est conforme à la spécification SQI (Simple Query Interface) [SMVA+05] et permet aux consommateurs de récupérer les données stockées dans le référentiel de contexte. Un des avantages de cette spécification est que le consommateur peut exprimer des requêtes en utilisant un langage qui convient le mieux à ses besoins. Il retrouve aussi les résultats dans un format approprié à ses objectifs.

- *Le service d'insertion* permet aux consommateurs d'indexer des données de contexte dans le référentiel. Ce service est compatible avec la spécification SPI (Simple Publishing Interface) et comme le service de recherche, il n'est pas spécifique à un certain format de métadonnées. Les consommateurs peuvent envoyer des données selon le format qui convient le mieux à leurs besoins.

- *Le service de gestion du modèle* permet de modifier et d'étendre les modèles de contexte brut quand il est nécessaire d'observer de nouvelles applications. Une partie des méthodes s'applique sur les classes, tandis que les autres s'appliquent sur les attributs. L'ensemble des méthodes comprend les transactions nécessaires à la gestion des données : insertion d'une nouvelle classe ou d'un nouvel attribut dans une classe existante, mise à jour d'une classe existante ou d'un attribut.

- *Le service des indicateurs* a un rôle double. Il permet aux applications tierces d'être informées (par principe d'abonnement/désabonnement) quand un ou plusieurs indicateurs d'intérêt sont calculés. Ce service accepte également des requêtes en provenance du notificateur des indicateurs composées de la définition et de la valeur de l'indicateur nouvellement calculé. Il fait ensuite suivre cette information à toutes les applications abonnées à cet indicateur.

La couche des applications d'apprentissage adaptatives

La couche AdWLE comprend des outils et des applications à partir desquels les métadonnées de contexte sont collectées et / ou réutilisées à des fins d'adaptation. Ces applications peuvent intégrer divers composants : (1) un capteur chargé de l'extraction des informations définies au sein de nos modèles de contexte, (2) un composant adaptatif qui interagit avec les deux couches adjacentes pour proposer des mécanismes d'adaptation différents, (3) un composant pour s'abonner et pour se désabonner aux indicateurs à travers le service des indicateurs, (4) un *listener* des indicateurs pour permettre aux applications de recevoir des notifications de la part du service des indicateurs, et (5) un concepteur de modèle graphique pour exprimer les besoins en terme de données de contexte.

La couche de l'utilisateur

Elle représente le dispositif utilisé par l'utilisateur pour accéder à l'ensemble des fonctionalités du système d'apprentissage. Cette couche permet de traiter la problematique liée à la protection de la vie privée en terme de collecte, traitement et stockage de données personnelles. Dans cet objectif nous avons introduit certains principes organisationnels ainsi que quelques directives à respecter par les applications utilisant notre cadre de travail. Les principes organisationnels proposent la séparation des données de contexte en deux espaces de stockage. Les données sensibles (le contexte de l'utilisateur) sont stockées sur le dispositif local tandis que les données non sensibles (le contexte de l'environnement

et de l'usage) sont stockées sur le référentiel central. L'authentification par pseudonyme est utilisée pour identifier l'utilisateur à la fois sur la machine locale et sur le référentiel central. Les directives décrivent les interactions entre les différents composants de notre architecture pour garantir le respect des sept principes de la vie privée.

PARTIE III - Implémentation

La troisième partie présente la mise en œuvre des modèles et de l'architecture exposés dans la Partie II et comprend trois chapitres. Le premier chapitre donne un aperçu des technologies et formats utilisés pour mettre en œuvre notre architecture, le deuxième chapitre présente quelques AdWLEs intégrés dans notre cadre de travail avec leurs modèles de contextes, alors que le troisième chapitre expose certaines applications et outils qui exploitent les données collectées selon deux cas d'usage.

Chapitre 6 - Les technologies et formats adoptés

Ce chapitre décrit les technologies que nous avons utilisées pour mettre en œuvre notre architecture ainsi que les formats adoptés par les services de la couche intermédiaire pour insérer, rechercher et récupérer des données contextuelles.

La couche des données contextuelles

Après avoir testé plusieurs types de système de stockage (une implémentation de WBEM native, une base de données XML et une base de données orientée objet) pour mettre en œuvre la couche de contexte, nous avons identifié la base de données Oracle Objet-Relationnel comme la solution la mieux adaptée. Elle offre un temps de réponse optimal aux requêtes complexes appliquées sur une grande quantité de données en combinant les avantages des paradigmes orientés objet et relationnels. Les données sont modélisées comme des objets et peuvent être facilement manipulées à travers le langage de requête SQL. Les composants responsables de la gestion des indicateurs sont mis en œuvre à travers le package Database Change Notification (DCN) d'Oracle qui permet aux objets référencés par des requêtes spécifiques d'être associés à une procédure de *callback* pour effectuer des traitements internes ou externes.

La couche intermédiaire

Les différents services de la couche intermédiaire ont été développés comme des services Web SOAP en utilisant le langage de programmation Java. En ce qui concerne les formats d'entrées / sorties

requis par ces services, nous avons choisi SQL comme langage de requête et SQL2XML comme format de résultat pour le service de recherche, ainsi que XML pour exprimer les métadonnées de contexte utilisées par le service d'insertion. SQL et SQL2XML apportent une flexibilité pour interroger n'importe quelle information stockée dans le référentiel, mais un schéma XML dynamique impose des contraintes sémantiques aux métadonnées de contexte collectées. Ce schéma est également flexible, car il propose des structures différentes pour chaque entité à observer en fonction des modèles de contexte. Nous avons également exploité le format standardisé xmlCIM pour représenter les extensions CIM des modèles de contexte ainsi que les instances CIM décrivant les définitions de nouveaux indicateurs.

La couche des applications d'apprentissage adaptatives

Les systèmes de la couche AdWLE qui ont été observés à ce jour sont les suivants : Moodle (la plateforme utilisée à l'Université de Toulouse), ARIADNE Finder (un moteur de recherche dédié à l'éducation qui donne accès à un ensemble de ressources pédagogiques réparties au niveau européen et mondial), et un outil web de questionnaires. Tous ces outils *open source* intègrent un capteur spécifique développé dans le langage natif de l'application cible, à savoir PHP pour Moodle et l'outil de questionnaires, javascript pour ARIADNE Finder. Un composant adaptatif a été intégré dans ARIADNE Finder pour recommander des objets pédagogiques à l'utilisateur en fonction de ses intérêts en cours. Le composant ainsi que le processus de recommandation sont détaillés dans le Chapitre 8. Un *listener* d'indicateurs a été integré dans Moodle en tant que service Web à des fins de validation. Ce *listener* reçoit des valeurs mises à jour d'un indicateur de proportion d'activités détaillé dans le chapitre suivant). L'abonnement correspondant a été effectué via le service des indicateurs.

La couche de l'utilisateur

Deux approches ont été mises en place pour stocker des données sensibles sur le dispositif de l'utilisateur : une implémentation native WBEM intégrée dans Windows (WMI - Windows Management Instrumentation) et des cookies. La première ne garantit pas la généricité de l'approche et n'est pas adaptée à une utilisation Web. En revanche, le stockage de données sensibles sur le dispositif local en utilisant les cookies présente plusieurs avantages : portabilité, confidentialité, mise à l'échelle et facilité d'utilisation. En outre, placer du code source sur le dispositif de l'utilisateur permet d'accéder, à partir de l'application WBEM native, aux informations des composants matériels et logiciels du dispositif. Cela enrichit le contexte de l'utilisateur et offre de nouvelles perspectives d'adaptation.

Chapitre 7 - Sources des métadonnées de contexte

Dans ce chapitre, nous mettons en avant deux caractéristiques essentielles de notre cadre de travail : (1) la capacité d'extension des modèles à représenter le contexte spécifique de nouvelles applications, et (2) la capacité du cadre de travail à définir et calculer des indicateurs sur la base du contexte brut collecté.

A partir des modèles de contextes génériques que nous avons présentés au Chapitre 4, les contextes de trois applications ont été modélisés : la plateforme Moodle utilisée à l'Université de Toulouse, l'outil Web de questionnaires du projet CONTINT, et l'application ARIADNE Finder.

La plate-forme Moodle de l'IUT A Paul Sabatier

La première application interagissant avec notre cadre de travail est la plate-forme Moodle déployée à l'IUT A Paul Sabatier. Cette plate-forme est entièrement intégrée dans le processus de formation traditionnel, et permet aux enseignants de présenter des ressources pédagogiques, des devoirs et diverses activités complémentaires. Parmi les éléments modélisés, on y trouve la plate-forme elle-même, les cours, les objets pédagogiques, les devoirs, les forums, ainsi que les activités spécifiques à chacun de ces éléments. Ces activités sont par exemple les connexions/déconnexions à la plate-forme, les consultations, les téléchargements et suppressions des cours, les publications de messages dans un fil de discussion d'un forum, ou encore les mises à jour ou suppressions de messages.

La supervision de la plate-forme a commencé début Janvier 2011, et sur la base des métadonnées collectées, nous avons défini des indicateurs élémentaires, arithmétiques et complexes. Parmi les indicateurs élémentaires nous avons calculé des informations statistiques telles que le nombre total de consultations, de téléchargements ou d'évaluations d'un objet pédagogique. Parmi les indicateurs complexes nous avons défini l'indicateur de proportion d'activités (PAI). Cet indicateur identifie le rôle joué par chaque participant lors d'un processus d'apprentissage collaboratif : il calcule la quantité d'activités réalisé par chaque utilisateur qui agit sur un ensemble de ressources. Puisque le PAI s'appuie sur un nombre variable d'indicateurs intermédiaires (par exemple, le nombre d'activités réalisées par chaque utilisateur sur chaque objet pédagogique d'un cours), il ne peut pas être calculé de la même manière qu'un indicateur élémentaire ou arithmétique. Sur la base de ce PAI, des outils réflexifs et de guidage peuvent être conçus.

Les questionnaires du projet CONTINT

Notre deuxième expérimentation a été menée dans le cadre du projet CONTINT. Ce projet, financé par l'Agence Nationale de la Recherche (ANR), vise à mettre en œuvre une démarche de qualité liée à un processus de réingénierie globale pour l'acquisition et le développement des aptitudes et des

compétences. Dans le cadre du projet, certaines expérimentations ont été menées sur une série de questionnaires développés sous forme de pages web interactives, intégrées au sein d'une plate-forme Moodle. Notre tâche a été de recueillir les interactions des utilisateurs avec ces questionnaires à l'aide de notre cadre de travail. Les données à collecter sur un tel questionnaire comprennent sa description, les questions et leurs propositions, ainsi qu'un ensemble d'activités associées (commencer ou finir un questionnaire, cocher une proposition, passer à la question suivante, etc.). Les expérimentations du projet sont réparties sur 4 années (2011 - 2014), et à ce jour huit séances d'expérimentation ont été observées à l'aide de notre cadre de travail.

Sur la base des métadonnées de contexte collectées, différents indicateurs ont été définis pour le projet CONTINT. A titre d'exemple, un des questionnaires s'appuie sur MSLQ (Motivated Strategies for Learning Questionnaire), un instrument d'auto-évaluation conçu pour évaluer les orientations motivationnelles des étudiants ainsi que leurs stratégies d'apprentissage [PDG90]. Plusieurs indicateurs métacognitifs ont ensuite été calculés : but de performance d'approche, sentiment d'auto efficacité, autocontrôle, planification, but de performance d'évitement, but de maîtrise, etc.

L'application ARIADNE Finder

La troisième application que nous avons observée est ARIADNE Finder qui fournit une recherche fédérée sur des viviers d'objets pédagogiques distribués. L'interface contient un champ de saisie où l'utilisateur introduit un ou plusieurs mots-clés. Une liste d'objets pédagogiques associés à ces mots-clés est ensuite présentée à l'utilisateur. Nous avons collecté des données à partir d'une instance locale de cette application. En raison d'une faible visibilité du serveur, les données recueillies ne sont pas significatives (quelques centaines d'activités seulement). Plusieurs indicateurs élémentaires ont été calculés, comme par exemple le nombre total d'objets pédagogiques indexés dans un vivier.

La vue unifiée des contextes des différentes applications permet d'améliorer le processus d'adaptation, car elle représente les activités des utilisateurs sur une multitude d'applications.

Chapitre 8 - Applications construites à partir de notre cadre de travail

Ce chapitre montre comment notre cadre de collecte et de partage de données contextuelles peut être utilisé pour réaliser des applications tierces. Nous avons développé deux applications : un service de recommandation personnalisée des objets pédagogiques, et un outil de visualisation.

Le service de recommandation des objets pédagogiques

Le premier outil présente une solution de recommandations des documents aux apprenants en fonction de leurs intérêts d'apprentissage courants et de leurs activités. Ce processus est possible grâce (1)

aux métadonnées de contexte recueillies lorsqu'un utilisateur effectue une activité sur une ressource, (2) aux modèles de l'utilisateur et du document décrits en termes d'annotations sémantiques, et (3) à un algorithme capable de calculer les similarités entre ces modèles.

Le profil de l'utilisateur est mis à jour chaque fois que l'utilisateur visite un document selon les annotations associées à ce dernier. L'algorithme de recommandation utilise une approche hybride. Un filtrage fondé sur le contenu entre le profil de l'utilisateur et la collection des documents permet de rechercher des documents susceptibles d'être intéressants pour l'utilisateur. A partir de ces résultats et du dernier document visité, un filtrage collaboratif permet de déterminer la liste des documents à recommander.

L'outil de visualisation MultiTravle

Le deuxième outil que nous avons développé est MultiTravle (Multi-touch TRAcking VisuaLization systEm), une application tactile de visualisation des données contextuelles. Cette application récupère les métadonnées de contexte à partir des services de la couche intermédiaire et les affiche de manière générique.

La visualisation est réalisée sur trois plans : (1) le plan des plates-formes de notre cadre de travail, localisées sur une carte du monde, (2) celui des ressources, et (3) celui des utilisateurs. Ces trois plans sont interconnectés et offrent une navigation récursive qui, à chaque étape, filtre l'affichage du plan suivant à partir de l'entité sélectionnée au plan précédent.

Le modèle de données de l'application s'appuie sur une représentation fidèle de notre modèle de contexte. A partir des entités sélectionnées par l'utilisateur, l'application crée des requêtes adéquates auprès du service de recherche pour récupérer les données qui seront affichées sur le plan suivant. Le langage de requête utilisé est SQL, tandis que le format des résultats est SQL2XML.

Les possibilités de construire d'autres applications ou services fondés sur notre cadre de travail restent ouvertes. Ainsi, au sein de notre équipe de recherche, une thèse a été initiée pour la construction d'un système de tutorat intelligent.

CONCLUSIONS

Chapitre 9 - Conclusions et perspectives de recherche

Les travaux présentés dans ce manuscrit ont relevés certains défis réels des AdWLEs existants [VMO+12], et ont mis l'accent en particulier sur le partage et la réutilisation des métadonnées de contexte à grande échelle. Pour atteindre ces objectifs, nos contributions portent sur trois propositions principales : (1) un modèle unificateur capable de fédérer les données provenant des AdWLEs hétérogènes, (2) une

architecture distribuée s'appuyant sur des interfaces standardisées et sur un couplage faible des composants, et (3) des principes organisationnels et lignes directrices qui assurent la confidentialité des données sensibles des acteurs de l'apprentissage en ligne.

Un modèle unificateur pour fédérer des données de contexte hétérogènes

Le modèle orienté objet que nous avons élaboré est fondé sur la norme CIM (Common Information Model), et comprend trois principaux sous-modèles interconnectés : (1) le contexte de l'utilisateur qui décrit les apprenants en termes de style d'apprentissage, de préférences ou de connaissances, (2) le contexte d'environnement qui décrit les systèmes adaptatifs ainsi que les ressources pédagogiques contenues par ces applications, et (3) le contexte d'usage qui décrit les actions entreprises par les utilisateurs sur les systèmes et les ressources. Notre modèle est caractérisé par une structure fixe de haut niveau d'abstraction qui constitue la base pour toutes les applications observées, mais permet également la définition d'informations détaillées et spécifiques à une application donnée ; cette modélisation offre ainsi un bon compromis entre généricité et facilité d'utilisation. De plus, nous avons intégré un modèle dédié à la définition et au calcul d'indicateurs sur la base des trois sous-modèles. Nous avons fait une distinction claire entre la définition de l'indicateur, sa valeur, et l'entité à laquelle il s'applique. Cette conception permet de définir précisément la signification des données inférées, de mettre en avant leur réutilisation pour d'autres entités, et de faciliter la conception d'indicateurs complexes.

Notre approche de modélisation a été validée par la fédération des métadonnées de contexte fournies par trois applications différentes : une plate-forme d'apprentissage Moodle, un outil de questionnaires, et ARIADNE Finder, un outil de recherche d'objets pedagogiques. Sur la base de métadonnées recueillies, nous avons calculé différents indicateurs aussi bien simples que complexes.

Comme perspectives de recherche liées au modèle, nous envisageons d'étudier comment des formats propriétaires comme ceux de l'initiative dataTEL[1] peuvent être fédérés à partir de notre modèle. Une autre perspective consiste à utiliser le modèle de politique CIM [DMT03] pour agréger des activités de différents niveaux d'abstraction.

Une architecture ouverte qui favorise le partage et la réutilisation des données contextuelles

Afin de soutenir nos modèles, nous avons adopté une architecture distribuée composée de quatre couches. La couche de contexte comprend un référentiel pour stocker les données et un ensemble de composants dédiés à la gestion des indicateurs. La couche intermédiaire offre un accès facile au référentiel de contexte. La couche AdWLE comprend les systèmes et outils de collecte des informations

[1] http://teleurope.eu/pg/groups/9405/datatel/

contextuelles. Enfin, la couche utilisateur représente le dispositif utilisé pour accéder aux systèmes d'apprentissage adaptatifs.

La couche intermédiaire s'appuie sur une architecture orientée services. Cette couche est actuellement composée de 4 services de base, standardisés et indépendants de tout format spécifique :

- Le service de recherche offre la possibilité de retrouver des données contextuelles au sein du référentiel,
- Le service d'insertion permet d'indexer les données de contexte collectées,
- Le service de gestion du modèle permet d'étendre les modèles de base,
- Le service des indicateurs permet aux applications de définir et de s'abonner aux indicateurs qui les intéressent.

Pour interagir avec ces services, nous avons étendu plusieurs systèmes d'apprentissages avec des composants différents. Certains d'entre eux sont responsables de la collecte des informations contextuelles du système (appelés capteurs), tandis que d'autres sont responsables de la manipulation du processus d'adaptation. La plateforme Moodle et un outil de questionnaires spécifiques, intègrent uniquement un capteur, tandis que l'application ARIADNE Finder intègre à la fois un capteur et un composant adaptatif pour recommander du contenu personnalisé aux utilisateurs. Enfin, deux propositions ont été mises en place pour implémenter la couche utilisateur.

Comme perspective nous voulons considérer les bases de données documentaires, pour gérer une importante quantité de données. Afin de rendre notre cadre de travail plus léger et plus rapide nous prévoyons de mettre en œuvre des services en utilisant l'architecture REST. Pour faciliter l'intégration de capteurs, nous prévoyons d'offrir des bibliothèques spécialisées destinées aux développeurs. Nous devons nous concentrer aussi sur les utilisateurs finaux pour offrir aux enseignants et apprenants, une interface graphique intuitive pour la définition et la réutilisation de données contextuelles. Enfin, nous voulons explorer des techniques de *Machine Learning* pour le calcul d'indicateurs.

Un cadre qui respecte la vie privée des utilisateurs

Afin d'assurer la confidentialité des utilisateurs dans notre cadre de travail, nous avons choisi de stocker les données sensibles (le contexte de l'utilisateur) sur le périphérique local, et les autres données (les contextes de l'environnement et d'usage) sur le référentiel central. Nous avons proposé deux solutions différentes pour le stockage des profils d'utilisateurs : une application WBEM intégrée nativement dans le système d'exploitation Windows, et l'utilisation des cookies. Ces derniers étant appropriés pour une utilisation sur le Web, cette approche a été choisie. En plus de ce principe organisationnel, nous avons

également introduit des exigences qui doivent être prises en compte par les différents composants de l'architecture pour respecter la législation relative à la vie privée des utilisateurs.

Le filtrage collaboratif est une des techniques les plus connues d'adaptation [Bur07]. Actuellement cette technique est limitée puisque les profils résident sur la machine cliente. Afin de supprimer cette limitation, certaines études doivent être menées pour identifier les composants du modèle de l'utilisateur qui peuvent être partagés sur le référentiel central sans compromettre leur identité. Ainsi, nous devons effectuer des analyses sur le contexte fourni par des sources multiples afin d'assurer que le recoupement des données ne permet pas de révéler l'identité d'un utilisateur.

Bibliography

[AAH+97] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A mobile context-aware tour guide. *Wireless Networks*, 3:421–433, 1997. Cited on page 36.

[AKD06] H. S. Al-Khalifa and H. C. Davis. The evolution of metadata from standards to semantics in e-learning applications. In *Proceedings of the 17th conference on Hypertext and hypermedia (HyperText 06)*, pages 69–72. ACM, 2006. Cited on pages 17 and 150.

[AKMF04] N. Avouris, V. Komis, M. Margaritis, and C. Fidas. Modellingspace: A tool for synchronous collaborative problem solving. In *World Conference on Educational Multimedia, Hypermedia and Telecommunications (EdMedia 04)*, number 1, pages 381–386, 2004. Cited on page 27.

[ASS+03] V. Aleven, E. Stahl, S. Schworm, F. Fischer, and R. Wallace. Help seeking and help design in interactive learning environments. *Review of Educational Research*, 73(3):277–320, 2003. Cited on pages 4 and 196.

[AT01] G. Adomavicius and A. Tuzhilin. Using data mining methods to build customer profiles. *Computer*, 34(2):74–82, 2001. Cited on page 24.

[AT11] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. 2011. Cited on page 36.

[AW05] I. Arroyo and B. Woolf. Inferring learning and attitudes from a bayesian network of log file data. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED 05)*, pages 33–40, 2005. Cited on page 164.

[BB05] M. Bazire and P. Brézillon. Understanding context before using it. *Modeling and Using Context*, pages 29–40, 2005. Cited on page 36.

[BBB+10] J. Broisin, M. Brut, V. Butoianu, F. Sedes, and P. Vidal. A personalized recommendation framework based on cam and document annotations. *Procedia Computer Science*, 1(2):2839–2848, 2010. Cited on page 149.

[BBK12] M. Bienkowski, J. Brecht, and J. Klo. The learning registry: building a foundation for learning resource analytics. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge (LAK 12)*, pages 208–211. ACM, 2012. Cited on page 56.

[BCQ+07] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. A data-oriented survey of context models. *ACM SIGMOD Record*, 36(4):19–26, 2007. Cited on page 66.

[BCVB11] V. Butoianu, O. Catteau, P. Vidal, and J. Broisin. Un système à base de traces pour la recherche personnalisée d'objets pédagogiques: le cas d'ariadne finder. In *Atelier "Personnalisation de l'apprentissage : quelles approches pour quels besoins ?" (EIAH 11)*, 2011. Cited on page 153.

[BDF+05] K. Borcea, H. Donker, E. Franz, A. Pfitzmann, and H. Wahrig. Towards privacy-aware elearning. In *Proceedings of 5th International Workshop on Privacy Enhancing Technologies (PET 05)*, volume 3856 of *Lecture Notes in Computer Science*, pages 167–178, 2005. Cited on page 108.

[BF00] B. Barros and M. Felisaverdejo. Analysing student interaction processes in order to improve collaboration. *International Journal of Artificial Intelligence in Education*, 11:221–241, 2000. Cited on pages xvii and 29.

[BHM06] C. J. Butz, S. Hua, and R. B. Maguire. A web-based bayesian intelligent tutoring system for computer programming. *Web Intelligence and Agent Systems*, 4(1):77–97, 2006. Cited on pages xvii, 30, 31, and 32.

[BL97] M. J. A. Berry and G. S. Linoff. *Data Mining Techniques. For Marketing, Sales, and Customer Support*. Wiley, 1997. Cited on page 36.

[Blo12] J. Blomer. Nsdl paradata. `https://wiki.ucar.edu/display/nsdldocs/comm_para`, 2012. Online, Accessed on 24 October 2012. Cited on page 58.

[BMPS09] E. Bertino, L. Martino, F. Paci, and A. Squicciarini. *Security for Web Services and Service-Oriented Architectures*. Springer Publishing Company, Incorporated, 2009. Cited on page 92.

[Bru98] P. Brusilovsky. Adaptive educational systems on the world wide web: A review of available technologies. In *Proceedings of "WWW-Based Tutoring" Workshop at 4th International Conference on Intelligent Tutoring Systems (ITS 98)*, 1998. Cited on pages 4 and 196.

[BSL+08] P. Brusilovsky, S. Sosnovsky, D.H. Lee, M. Yudelson, V. Zadorozhny, and X. Zhou. An open integrated exploratorium for database courses. In *Proceedings of 13th International Conference on Innovation and Technology in Computer Science Education (ITiCSE 08)*, 2008. Cited on pages xvii and 50.

[Bur07] R. Burke. Hybrid web recommender systems. *The adaptive web*, pages 377–408, 2007. Cited on pages 165 and 212.

[BV05] J. Broisin and P. Vidal. Un environnement informatique pour l'apprentissage humain au service de la virtualisation des objets pédagogiques. *Revue Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation*, 12, 2005. Cited on page 20.

[BVB09] V. Butoianu, P. Vidal, and J. Broisin. An adaptative framework for tracking web-based learning environments. In *Exploitation of Usage of Attention Metadata Workshop (EUAM 09)*, 2009. Cited on page 100.

[BVB11a] M. El A. Bouabid, P. Vidal, and J. Broisin. A web application dedicated to online practical activities: the case of system and network experiments. In *IEEE International Conference on Advanced Learning Technologies (ICALT 11)*, pages 93–97. IEEE Computer Society, 2011. Cited on page 6.

[BVB11b] V. Butoianu, P. Vidal, and J. Broisin. Partage d'un corpus de données d'observation issues d'activités d'apprentissage. In *Atelier "Partager des données d'observation pour la recherche en EIAH traces d'activité d'apprentissage" (EIAH 11)*, 2011. Cited on page 135.

[BVB11c] V. Butoianu, P. Vidal, and J. Broisin. Prise en compte de la vie privée des usagers dans un système à base de traces dédié à l'apprentissage en ligne. In *Environnements Informatiques pour l'Apprentissage Humain (EIAH 11)*, pages 355–367. Association des Technologies de l'Information pour l'Education et la Formation (ATIEF), 2011. Cited on page 108.

[BVB11d] V. Butoianu, P. Vidal, and J. Broisin. A recommendation algorithm based on documents titles and dynamic changes of learners interests. In *International Conference on Remote Engineering and Virtual Instrumentation (REV 11)*, page electronic medium. International Association of Online Engineering, 2011. Cited on page 153.

[BVB12] V. Butoianu, P. Vidal, and J. Broisin. A model-driven approach to actively manage tel indicators. In *World Conference on Educational Multimedia, Hypermedia and Telecommunications (EdMedia 12)*, volume 2012, pages 1757–1765, 2012. Cited on page 83.

[BVV+10] V. Butoianu, P. Vidal, K. Verbert, E. Duval, and J. Broisin. User context and personalized learning: a federation of contextualized attention metadata. *Journal of Universal Computer Science*, 16(16):2252–2271, 2010. Cited on pages 39 and 73.

[CBY11] R. C. Chen, C. T. Bau, and C. J. Yeh. Merging domain ontologies based on the wordnet system and fuzzy formal concept analysis techniques. *Applied Soft Computing*, 11(2):1908–1923, 2011. Cited on page 75.

[CDL04] C. M. Chen, L. J. Duh, and C. Y. Liu. A personalized courseware recommendation system based on fuzzy item response theory. In *IEEE International Conference on e-Technology, e-Commerce, and e-Services (EEE 05)*, pages 305–308. IEEE Computer Society, 2004. Cited on page 33.

[CGCC06] W. F. Contreras, E. G. Galindo, E. M. Caballero, and G. M. Caballero. An intelligent tutoring system for a virtual e-learning center. *Current Developments in Technology Assisted Education*, pages 768–772, 2006. Cited on pages 30 and 198.

[CGME02] N. Capuano, M. Gaeta, A. Micarelli, and Sangineto. E. An integrated architecture for automatic course generation. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT 02)*, pages 322–326, 2002. Cited on pages xvii, 25, and 31.

[CHB09] M. Cocea, A. Hershkovitz, and R. S. J. Baker. The impact of off-task and gaming behaviors on learning: immediate or aggregate? 2009. Cited on page 164.

[CHSW04] W. C. Chang, H. H. Hsu, T. K. Smith, and C. C. Wang. Enhancing scorm metadata for assessment authoring in e-learning. *Journal of Computer Assisted Learning*, 20(4):305–316, 2004. Cited on page 19.

[CI06] C. Choquet and S. Iksal. Usage tracking language: a meta language for modelling tracks in tel systems. In *International Conference on Software Technologies (ICSOFT 06)*, pages 133–138, 2006. Cited on page 53.

[CI07] C. Choquet and S. Iksal. Modeling tracks for the model driven reengineering of a tel system. *Journal of Interactive Learning Research, Special Issue: Usage Analysis in Learning Systems Existing Approaches and Scientific Issues*, 18(2):161–184, 2007. Cited on pages xviii and 54.

[Com95] European Commission. Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with the regard to the processing of personal data and on the free movement of such data. *Official journal of the European Communities*, (L. 281):31, 1995. Cited on pages 7, 44, and 200.

[Com11] European Commission. Attitudes on data protection and electronic identity in the european union. `http://ec.europa.eu/public_opinion/archives/ebs/ebs_359_en.pdf`, 2011. Online, Accessed on 24 October 2012. Cited on page 44.

[CPFJ04] H. Chen, F. Perich, T. Finin, and A. Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS 04)*, 2004. Cited on page 42.

[CW07] M. Cocea and S. Weibelzahl. Eliciting motivation knowledge from log files towards motivation diagnosis for adaptive systems. *User Modeling*, pages 197–206, 2007. Cited on page 164.

[CW09] M. Cocea and S. Weibelzahl. Log file analysis for disengagement detection in e-learning environments. *User Modeling and User-Adapted Interaction*, 19(4):341–385, 2009. Cited on page 164.

[Dav02] J. Davis. Introduction to cim-xml. `http://members.dmtf.org/data/presentations/devcon02/JimDavis-IntroductiontoCIM-XML.pdf`, 2002. Online, Accessed on 11 February 2013. Cited on page 123.

[DBAC04a] P. De Bra, L. Aroyo, and V. Chepegin. The next big thing: Adaptive web-based systems. *Journal of Digital Information*, 5(1), 2004. Cited on pages 22 and 23.

[DBAC04b] P. De Bra, L. Aroyo, and A. Cristea. Adaptive web-based educational hypermedia. *Web dynamics, adaptive to change in content, size, topology and use*, pages 387–410, 2004. Cited on page 23.

[DBHW99] P. De Bra, G. J. Houben, and H. Wu. Aham: A dexter-based reference model for adaptive hypermedia. In *Proceedings of the tenth ACM Conference on Hypertext and hypermedia: returning to our diverse roots (HyperText 99)*, pages 147–156, 1999. Cited on page 26.

[DDJ+05] A. N. Dragunov, T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker. Tasktracer: a desktop environment to support multi-tasking knowledge workers. In *Proceedings of the 10th international conference on Intelligent user interfaces (IUI 05)*, pages 75–82. ACM, 2005. Cited on pages xvii, 46, and 47.

[Dey01] A. K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5:4–7, 2001. Cited on page 37.

[Dim04] A. Dimitrakoupoulou. D26.1.1: State of the art on interaction and collaboration analysis. In *Kaleidoscope Deliverables*, 2004. Cited on page 26.

[Djo11] T. Djouad. *Ingénierie des indicateurs d'activités à partir de traces modélisées pour un Environnement Informatique d'Apprentissage Humain*. PhD thesis, Université Claude Bernard - Lyon I, 2011. Cited on pages xvii and 27.

[DL05] P. C. David and T. Ledoux. Wildcat: a generic framework for context-aware applications. In *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing (MPAC 05)*, pages 1–7. ACM, 2005. Cited on pages xvii and 49.

[DMT03] DMTF. Cim policy model white paper. `http://dmtf.org/sites/default/files/standards/documents/DSP0108.pdf`, 2003. Online, Accessed on 18 February 2013. Cited on pages 162 and 210.

[DSP+10] T. Djouad, L. S. Settouti, Y. Prié, C. Reffay, and A. Mille. Un système à base de traces pour la modélisation et l'élaboration d'indicateurs d'activités éducatives individuelles et collectives. mise à l'épreuve sur moodle. *Technique et Science Informatiques*, 29(6):721–741, 2010. Cited on pages xviii, 55, 56, and 63.

[EHLS06] M. Engelhardt, A. Hildebrand, D. Lange, and T. C. Schmidt. Reasoning about elearning multimedia objects. In *International Workshop on Semantic Web Annotations for Multimedia (SWAMM 06)*. Springer, 2006. Cited on page 150.

[FC00] N. Ford and S. Y. Chen. Individual differences, hypermedia navigation, and learning: an empirical study. *Journal of educational multimedia and hypermedia*, 9:281–311, 2000. Cited on page 23.

[Fed00] P. A. Federico. Learning styles and student attitudes toward various aspects of network-based instruction. *Computers in Human Behavior*, 16(4):359–379, 2000. Cited on page 5.

[FH95] R. M. Felder and E. R. Henriques. Learning and teaching styles in foreign and second language education. *Foreign Language Annals*, 28(1):21–31, 1995. Cited on page 23.

[FH05] M. Feng and N. T. Heffernan. Informing teachers live about student learning: Reporting in the assistment system. In *the 12th Annual Conference on Artificial Intelligence in Education, Workshop on Usage Analysis in Learning Systems (AIED 05)*, 2005. Cited on page 28.

[FHMC07] L. France, J. M. Heraud, J. C. Marty, and T. Carron. Visualisation et régulation de l'activité des apprenants dans un eiah tracé. In *Environnements Informatiques pour l'Apprentissage Humain (EIAH 07)*, pages 197–184, 2007. Cited on page 28.

[FM95] B. Flinn and H. Maurer. Levels of anonymity. *Journal of Universal Computer Science*, 1(1):35–47, 1995. Cited on page 45.

[Hec05] D. Heckmann. *Ubiquitous User Modeling*, volume 297. IOS Press, 2005. Cited on pages xvii and 40.

[HI06] K. Henricksen and J. Indulska. Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing*, 2:37–64, 2006. Cited on pages xvii and 43.

[HN08] A. Hershkovitz and R. Nachmias. Developing a log-based motivation measuring tool. In *Proceedings of the International Conference on Educational Data Mining (EDM 08)*, pages 226–233, 2008. Cited on page 164.

[Hob04] C. Hobbs. *A Practical Approach to WBEM/CIM Management*. CRC, 1st edition, 2004. Cited on pages xviii and 88.

[Hod02] H. W. Hodgins. The future of learning objects. In *e-Technologies in Engineering Education: Learning Outcomes Providing Future Possibilities (eTEE 02)*, volume P1, pages 76–82. Corradini Eds, 2002. Cited on page 15.

[JD08] P. Jermann and P. Dillenbourg. Group mirrors to support interaction regulation in collaborative problem solving. *Computers & Education*, 51(1):279–296, 2008. Cited on page 28.

[JJG07] Z. Jeremic, J. Jovanovic, and D. Gasevic. Evaluating an intelligent tutoring system for design patterns: the depths experience. *Educational Technology Society*, pages 111–130, 2007. Cited on page 30.

[JR11] P. Jesukiewicz and D.R. Rehak. The learning registry: Sharing federal learning resources. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC 11)*, volume 2011, 2011. Cited on pages xviii and 57.

[KM04] R. Koper and J. Manderveld. Educational modelling language: modelling reusable, interoperable, rich and personalised units of learning. *British Journal of Educational Technology*, 35(5):537–551, 2004. Cited on page 19.

[Kob01a] A. Kobsa. Generic user modeling systems. *User Modeling and User-Adapted Interaction*, 11:49–63, 2001. Cited on page 42.

[Kob01b] A. Kobsa. Tailoring privacy to users' needs. In *Proceedings of the 8th International Conference on User Modeling (UM 01)*, pages 303–313. Springer-Verlag, 2001. Cited on page 45.

[KPP04] S. Kotsiantis, C. Pierrakeas, and P. Pintelas. Predicting students'performance in distance learning using machine learning techniques. *Applied Artificial Intelligence*, 18(5):411–426, 2004. Cited on page 164.

[LB11] Y. L. Lin and P. Brusilovsky. Towards open corpus adaptive hypermedia: a study of novelty detection approaches. In *Proceedings of the 19th International Conference on User Modeling, Adaption, and Personalization (UMAP 11)*, pages 353–358. Springer-Verlag, 2011. Cited on page 20.

[LBKL09] S. N. Lindstaedt, G. Beham, B. Kump, and T. Ley. Getting to know your user — unobtrusive user model maintenance within work-integrated learning environments. In *Proceedings of the 4th European Conference on Technology Enhanced Learning: Learning in the Synergy of Multiple Disciplines (EC-TEL 09)*, pages 73–87. Springer-Verlag, 2009. Cited on pages xvii and 53.

[LBM03] Y. Li, Z.A. Bandar, and D. Mclean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882, 2003. Cited on page 152.

[LCM+10] A. M. Latham, K. A. Crockett, D. A. McLean, B. Edmonds, and K. O'Shea. Oscar: An intelligent conversational agent tutor to estimate learning styles. In *IEEE International Conference on Fuzzy Systems (FUZZ 10)*, pages 1–8, 2010. Cited on page 32.

[Lej04] A. Lejeune. Ims learning design. *Distances et savoirs*, 2:409–450, 2004. Cited on page 19.

[LFBG07] R. Lokaiczyk, A. Faatz, A. Beckhaus, and M. Goertz. Enhancing just-in-time e-learning through machine learning on desktop context sensors. In *Proceedings of the 6th international and interdisciplinary conference on Modeling and using context (CONTEXT 07)*, pages 330–341. Springer-Verlag, 2007. Cited on pages xvii and 52.

[LIP08] IMS LIP. Ims learner information package specification. `http://www.imsglobal.org/profiles/index.html`, 2008. Online, Accessed on 17 April 2013. Cited on page 74.

[LM06] S. Lindstaedt and H. Mayer. A storyboard of the aposdle vision. *Innovative Approaches for Learning and Knowledge Sharing*, pages 628–633, 2006. Cited on page 52.

[LR07] R. Lanzilotti and T. Roselli. An experimental evaluation of logiocando, an intelligent tutoring hypermedia system. *International Journal of Artificial Intelligence in Education*, 17:41–56, 2007. Cited on page 32.

[LS01] C. Lagoze and Van De H. Sompel. The open archives initiative: building a low-barrier interoperability framework. In *Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries (JCDL 01)*, pages 54–62, 2001. Cited on page 17.

[Mei06] W. Meier. Index-driven xquery processing in the exist xml database. In *IXML Prague: a conference on XML*, 2006. Cited on page 117.

[MK07] E. McIlvain and M. Khoo. Nsdl privacy policy. `http://nsdlnetwork.org/sites/default/files/NSDL_Privacy_policy.pdf`, 2007. Online, Accessed on 27 October 2012. Cited on page 65.

[MLDLR03] M. Montaner, B. López, and J. L. De La Rosa. A taxonomy of recommender agents on the internet. *Artificial intelligence review*, 19:285–330, 2003. Cited on page 24.

[MM04] R. Mazza and C. Milani. Gismo: a graphical interactive student monitoring tool for course management systems. In *International Conference on Technology Enhanced Learning (TEL 04)*, pages 18–19, 2004. Cited on page 28.

[MM10] K. Mishra and R. B. Mishra. An intelligent tutoring system for c++. In *International Conference On Electronics and Information Engineering (ICEIE 10)*, volume 2, pages 454–458, 2010. Cited on page 31.

[MPG03] G. Magoulas, K. Papanikolaou, and M. Grigoriadou. Adaptive web-based learning: accommodating individual differences through system's adaptation. *British Journal of Educational Technology*, 34(4):200–3, 2003. Cited on pages xvii, 23, and 24.

[MSM07] A. Micarelli, F. Sciarrone, and M. Marinilli. Web document modeling. In *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 155–192. Springer, 2007. Cited on page 150.

[Mur99] T. Murray. Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10:98–129, 1999. Cited on page 31.

[ND08] L. Nguyen and P. Do. Learner model in adaptive learning. *Engineering and Technology*, 35:396–402, 2008. Cited on page 23.

[Neg98] M. Negnevitsky. A knowledge based tutoring system for teaching fault analysis. *IEEE Transactions on Power Systems*, 13(1):40–45, 1998. Cited on page 30.

[NIC10] D. P. T. Ngoc, S. Iksal, and C. Choquet. Re-engineering of pedagogical scenarios using the data combination language and usage tracking language. In *IEEE 10th International Conference on Advanced Learning Technologies (ICALT 10)*, pages 506–510, 2010. Cited on pages 27, 28, and 53.

[NICK09] D. P. T. Ngoc, S. Iksal, C. Choquet, and E. Klinger. Utl-cl: A declarative calculation language proposal for a learning tracks analysis process. In *9th IEEE International Conference on Advanced Learning Technologies (ICALT 09)*, pages 681–685, 2009. Cited on pages xviii, 53, and 54.

[NIS04] NISO. *Understanding Metadata*. National Information Standards Organization Press, 2004. Cited on page 16.

[NSW12] K. Niemann, M. Scheffel, and M. Wolpers. An overview of usage data formats for recommendations in tel. In *Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 12)*, pages 95–100, 2012. Cited on pages xviii, 57, and 58.

[ÖA09] E. Özpolat and G. B. Akar. Automatic detection of learning styles for an e-learning system. *Computers & Education*, 53(2):355–367, 2009. Cited on page 164.

[OR00] J. Ong and S. Ramachandran. Intelligent tutoring systems: The what and the how. *ASTD Learning Circuits*, 2000. Cited on page 30.

[Pau02] M. F. Paulsen. Online education systems: Discussion and definition of terms. `http://www.porto.ucp.pt/open/curso/modulos/doc/Definition%20of%20Terms.pdf`, 2002. Online, Accessed on 01 February 2013. Cited on page 20.

[PDG90] P. R. Pintrich and E. V. De Groot. Motivational and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology*, 82(1):33–40, 1990. Cited on pages 142 and 208.

[Per04] J.-P. Pernin. Lom, scorm et ims-learning design : ressources, activités et scenarios. *Paper presented at L'indexation des ressources pedagogiques numeriques (journee d'etude du 16/11/2004)*, 2004. Cited on page 19.

[PGG09] A. Papadimitriou, M. Grigoriadou, and G. Gyftodimos. Interactive problem solving support in the adaptive educational hypermedia system mathema. *IEEE Transactions on Learning Technologies*, 2:93–106, 2009. Cited on page 29.

[PLR04] A. Paramythis and S. Loidl-Reisinger. Adaptive learning environments and elearning standards. *Electronic Journal of Elearning*, 2:181–194, 2004. Cited on pages 20 and 21.

[PTG08] C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE Transactions on Knowledge and Data Engineering*, 20(11):1535–1549, 2008. Cited on page 36.

[PV05] Y. Peter and T. Vantroys. The ariadne infrastructure for managing and storing metadata. *Educational Technology Society*, 8(3):122–137, 2005. Cited on page 18.

[RBD+08] M. M. Rodrigo, R. S. Baker, S. D'Mello, M. C. Gonzalez, M. C. Lagud, S. A. Lim, A. F. Macapanpan, S. A. Pascua, J. Q. Santillano, J. O. Sugay, S. Tep, and N. J. Viehland. Comparing learners' affect while using an intelligent tutoring system and a simulation problem solving game. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS 08)*, pages 40–49. Springer-Verlag, 2008. Cited on page 30.

[RDL09] A. S. Rath, D. Devaurs, and S. N. Lindstaedt. UICO: an ontology-based user interaction context model for automatic task detection on the computer desktop. In *Workshop on Context, Information and Ontologies (ESWC 09)*, 2009. Cited on pages xvii, 38, and 51.

[RKL+08] A. Rath, M. Kröll, S. Lindstaedt, N. Weber, M. Granitzer, and O. Dietzel. Context-aware knowledge services. In *Proceedings of Computer Human Interaction (CHI 08), Workshop on Personal Information Management: (PIM 08)*, 2008. Cited on pages xvii and 52.

[RLSB08] M. Rosen, B. Lublinsky, K. T. Smith, and M. J. Balcer. *Applied SOA: Service-Oriented Architecture and Design Strategies*. Wiley Publishing, 2008. Cited on page 92.

[RVB10] M. T. Ramandalahy, P. Vidal, and J. Broisin. An intelligent tutoring system supporting metacognition and sharing learners' experiences. In *Proceedings of the 10th International Conference on Intelligent Tutoring Systems (ITS 10)*, pages 402–404. Springer, 2010. Cited on pages 74, 75, and 201.

[RvRK+02] A. Rawlings, P. van Rosmalen, R. Koper, M. Rodríguez-Artacho, and P. Lefrere. Survey of Educational Modelling Languages (EMLs) Version 1. `http://www.cenorm.be/cenorm/businessdomains/businessdomains/isss/activity/emlsurveyv1.pdf`, 2002. Online, Accessed on 21 October 2012. Cited on page 19.

[Sch08] R. Schapire. Cos 511: Theoretical machine learning: Lecture 1. `http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf`, 2008. Online, Accessed on 04 February 2013. Cited on page 164.

[SDVC+07] M. Stefaner, E. Dalla Vecchia, M. Condotta, M. Wolpers, M. Specht, S. Apelt, and E. Duval. Mace–enriching architectural learning objects for experience multiplication. In *Creating New Learning Experiences on a Global Scale*, volume 4753 of *Lecture Notes in Computer Science*, pages 322–336. Springer, 2007. Cited on page 48.

[SEBAM10] A. A. Saleh, H. M. El-Bakry, T. T. Asfour, and N. Mastorakis. Adaptive e-learning tools for numbering systems. In *Proceedings of the 9th WSEAS international conference on Applications of computer engineering (ACE 10)*, pages 293–298. World Scientific and Engineering Academy and Society, 2010. Cited on page 20.

[SFJ+09] M. Scheffel, M. Friedrich, M. Jahn, U. Kirschenmann, K. Niemann, H.-C. Schmitz, and M. Wolpers. Self-monitoring for computer users. In *GI Jahrestagung*, pages 1680–1687, 2009. Cited on page 49.

[Sim02] C. Simard. Normalisation de la formation en ligne: Enjeux, tendances et perspectives. *prepared for Agence Universitaire de la Francophonie(AUF)*, 2002. Cited on page 16.

[SK02] D. Sampson and C. Karagiannidis. Personalised learning: Educational, technological and standardisation perspective. *Interactive Educational Multimedia*, 4:24–39, 2002. Cited on page 30.

[SKKR02] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *5th International Conference on Computer and Information Science (ICIS 02)*, pages 27–28, 2002. Cited on page 150.

[SLP04] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management, at the 6th International Conference on Ubiquitous Computing (UbiComp 04)*, 2004. Cited on page 40.

[SM86] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986. Cited on page 152.

[SMJM05] A. Soller, A. Martínez, P. Jermann, and M. Muehlenbrock. From mirroring to guiding: A review of state of the art technology for supporting collaborative learning. *International Journal of Artificial Intelligence in Education*, 15(4):261–290, 2005. Cited on pages 27 and 28.

[SMVA+05] B. Simon, D. Massart, F. Van Assche, S. Ternier, E. Duval, S. Brantner, D. Olmedilla, and Z. Miklos. *A Simple Query Interface for Interoperable Learning Repositories*, pages 11–18. CEUR, 2005. Cited on pages 17, 49, 95, and 203.

[SPV07] K. Stefanidis, E. Pitoura, and P. Vassiliadis. A context-aware preference database system. *International Journal of Pervasive Computing and Communications*, 3(4):439–460, 2007. Cited on page 36.

[ST03] V. Shute and B. Towle. Adaptive e-learning. *Educational Psychologist*, 38:105–114, 2003. Cited on pages xvii, 5, 22, and 25.

[Stu09] L. W. Stuart. Dublin core metadata initiative: A personal history. *In Encyclopedia of Library and Information Science*, 3, 2009. Cited on page 16.

[SZR12] V. J. Shute and D. Zapata-Rivera. Adaptive educational systems. In *Adaptive technologies for training and education*, pages 7–27. Cambridge University Press, 2012. Cited on page 20.

[TCLW06] K. H. Tsai, T. K. Chiu, M. C. Lee, and T. I. Wang. A learning objects recommendation model based on the preference and ontological approaches. In *Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies (ICALT 06)*, pages 36–40. IEEE Computer Society, 2006. Cited on pages 26 and 33.

[TM05] T. Tang and G. McCalla. Smart recommendation for an evolving e-learning system: Architecture and experiment. *International Journal on E-Learning*, 4(1):105–129, 2005. Cited on page 32.

[TMVA+08] S. Ternier, D. Massart, F. Van Assche, N. Smith, B. Simon, and E. Duval. A simple publishing interface for learning object repositories. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (EdMedia 08)*, pages 1840–1845. AACE, June 2008. Cited on pages 17, 49, and 98.

[TVP+09] S. Ternier, K. Verbert, G. Parra, B. Vandeputte, J. Klerkx, E. Duval, V. Ordoez, and X. Ochoa. The ariadne infrastructure for managing and storing metadata. *IEEE Internet Computing*, 13(4):18–25, 2009. Cited on pages xvii, 17, 18, and 48.

[VMO+12] K. Verbert, N. Manouselis, X. Ochoa, M. Wolpers, H. Drachsler, I. Bosnic, and E. Duval. Context-aware recommender systems for learning: A survey and future challenges. *IEEE Transactions on Learning Technologies*, 5(4):318–335, 2012. Cited on pages 161 and 209.

[Wea12] J. Weatherley. Nsdl technical platform. `https://wiki.ucar.edu/display/nsdldocs/Technical+Overview`, 2012. Online, Accessed on 24 October 2012. Cited on pages xviii and 59.

[WNVD07] M. Wolpers, J. Najjar, K. Verbert, and E. Duval. Tracking actual usage: the attention metadata approach. In *International Journal of Educational Technology and Society*, pages 1176–3647. Press, 2007. Cited on pages xvii, 47, and 48.

[XP01] Y. Xie and V. V. Phoha. Web user clustering from access log using belief function. In *Proceedings of the 1st International Conference on Knowledge Capture (K-CAP 01)*, pages 202–208. ACM, 2001. Cited on page 24.

[Zai01] O. R. Zaiane. Web usage mining for a better web-based learning. In *Proceedings of Conference on Advanced Technology for Education (ATE 01)*, pages 60–64, 2001. Cited on page 4.

[Zai02] O. R. Zaiane. Building a recommender agent for e-learning systems. In *Proceedings of the International Conference on Computers in Education (ICCE 02)*, volume 1, pages 55–59. IEEE Computer Society, 2002. Cited on pages 33 and 195.

[ZLO07] A. Zimmermann, L. Lorenz, and R. Oppermann. An operational definition of context. *CONTEXT*, 2007. Cited on pages xvii, 37, and 38.