*Journal of Universal Computer Science, vol. 16, no. 16 (2010), 2252-2271*
*submitted: 31/3/10, accepted: 31/7/10, appeared: 28/8/10 © J.UCS*

# User Context and Personalized Learning: a Federation of Contextualized Attention Metadata

**Valentin Butoianu**
(Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier, France
butoianu@irit.fr)

**Philippe Vidal**
(Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier, France
vidal@irit.fr)

**Katrien Verbert**
(Department of Computer Science, K.U. Leuven, Belgium
Katrien.verbert@cs.kuleuven.be)

**Erik Duval**
(Department of Computer Science, K.U. Leuven, Belgium
erik.duval@cs.kuleuven.be)

**Julien Broisin**
(Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier, France
broisin@irit.fr)

**Abstract:** Nowadays, personalized education is a very hot topic in technology enhanced learning (TEL) research. To support students during their learning process, the first step consists in capturing the context in which they evolve. Users typically operate in a heterogeneous environment when learning, including learning tools such as Learning Management Systems and non-learning tools and services such as e-mails, instant messaging, or web pages. Thus, user attention in a given context defines the Contextualized Attention Metadata (CAM). Various initiatives and projects allow capturing CAMs in a knowledge workers' environment not only in the TEL area, but also in other domains like Knowledge Work Support, Personal Information Management and Information Retrieval. After reviewing main existing approaches according to some specific criteria that are of main interest for capturing and sharing user contexts, we present in this paper a framework able to gather CAMs produced by any tool or computer system. The framework is built on the Web-Based Enterprise Management (WBEM) standard dedicated to system, network and application management. Attention information specific to heterogeneous tools are represented as a unified and extensible structure, and stored into a central repository compliant with the above-mentioned standard. To facilitate access to this attention repository, we introduced a middleware layer composed of two dynamic services: the first service allows users to define the attention data they want to collect, whereas the second service is dedicated to receive and retrieve the traces produced by computer systems. An implementation for collecting and storing CAM data generated by the Ariadne Finder and Moodle validates our approach.

**Keywords:** Technology Enhanced Learning, Contextualized Attention Metadata
**Categories:** L.2.0, L.2.2, L.3.0, L.3.6

## 1 Introduction

Nowadays, personalized education is a very hot topic in technology enhanced learning (TEL) research. Learners are different in age, sex and social role, culture, education background, way of learning, knowledge, attention and interests. It is of vital importance to provide them with learning contents and teaching tactics according to their individual needs. To support students during their learning process, we need to capture the context in which they operate. The most generally accepted definition of context in the community is given by [Dey, 01]:*"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"*. We specially focus on the interactions of the user with applications and resources integrated within these applications, namely user interaction context. Current Learning Management Systems (LMS) such as Moodle [Moodle, 02] or Blackboard [Blackboard, 97] support the user in specific tasks (e.g. acquiring a skill) in specific environments. These LMSs are a good source of users' contextual information in specific learning environments (e.g. which learning material a learner is reading and which learning resources remain to be read). However, the user typically operates in a much larger environment while learning, using non-learning tools and services like e-mail, instant messaging, web pages, locally stored files and folders, etc. Capturing such information significantly extends the operating context of the user. Enhancing this information with data about the attention that users spend on learning resources (e.g. time spent to read a document, movement of the eyes, mouse clicks, scrolls) in a specific context will provide valuable additional information. Contextualized Attention Metadata (CAM) [Wolpers, 07] encapsulates such user attention in a given context. CAM provides a schema and a framework that capture users' interactions with different applications in a generalized format that allows merging and processing of streams of observations.

There are various approaches for capturing CAM in a user environment in different areas, such as Techonlogy Enhanced Learning (TEL) (CAM [Wolpers, 07], Aposdle [Lindstaedt, 06], Lip [Schmidt, 07]), Knowledge Work Support (Swish [Oliver, 06], TaskTracer [Dragunov, 05]), Personal Information Management (DYONIPOS [Rath, 09], GNOWSIS [Sauermann, 06], NEPOMUK [Groza, 08]) and Information Retrieval [Shen, 05] [Sugiyama, 04]. Even if these approaches present some advantages they give rise to some inconveniences therefore, no approach fulfills the entire set of important criteria that context-aware applications have to deal with. In this paper we present a framework that takes into account the advantages of presented approaches and that tackles their limitations. Our framework represents the first step toward personalization. Indeed, personalization tools (intelligent tutoring system, recommender system, etc) need CAM to achieve their task.

The paper is structured as follows: section 2 briefly describes some of the existing approaches regarding user context models and their associated architectures. Section 3 presents important requirements that context-aware applications should follow and how each approach respects them or not. Since there is no approach to apply them all, section 4 describes our complete framework for modeling and tracking usage data and how each requirement is satisfied (section 5). Finally, conclusions and future works are provided in section 6.

## 2 Existing approaches

Various approaches capture CAM in a knowledge workers' environment and use different representation formats for modeling the context of the user: key-value models, mark-up scheme models, graphical models, object oriented models, logic based models and ontology based models. While early models mainly addressed the modeling of context regarding one application, generic context models are of interest since multiple applications can benefit from these. A detailed survey of most relevant context models is presented in [Strang, 04], [Baldauf, 07] and [Bolchini, 07].

Approaches to implement CAM applications depend on special requirements as the location of sensors (local or remote), the amount of possible users (one or many), the available resources of the used device, or the facility of a further extension of the system. The context-data acquisition is very important when designing context-aware applications because it predefines the architectural style of the system. [Chen, 04] presented three different approaches on how to acquire contextual information:

- *Direct sensor access*, used in applications with sensors locally built in. The client application gathers information directly from the sensors. There is no additional layer for gaining and processing sensor data.
- *Middleware infrastructure* introduces a layered architecture (which includes a storage component) to context-aware applications that separates the collecting and processing phases. Compared with direct sensor access, this technique eases extensibility since the client code does not have to be modified anymore.
- *Context server* is a distributed approach which extends the middleware based architecture by introducing remote access to permit multiple clients to distantly retrieve encapsulated context data.

Since a separation of detecting and using context is necessary to improve extensibility and reusability of systems, the layered middleware and context-server systems are preferred. Henricksen's [Henricksen, 06] six layered architecture is one of the most comprehensive architectural approach:

1. *Gathering layer.* The lower-most layer consists of a collection of different sensors and interpreters/aggregators that capture information related to the content from the user environment.
2. *Reception layer.* This layer provides an interface between the gathering layer and the management layer. It translates data coming from the gathering layer into the model format described into the storage layer.
3. *Storage/Management layer* is responsible for storing context data in a context repository and for keeping it consistent.
4. *Query layer* adds query functionality to the repository.
5. *Adaptation layer* is responsible for encapsulating the adaptation logic for the application layer.
6. *Application layer* is composed by the context-aware applications which exploit the collected information in order to self-adapt to the current context.

In the next section, we present several approaches dealing with user context. For each approach, we briefly present on the one hand the model used to represent CAM as well as collected data, and one the other hand, how each approach implements each layer of the above architecture. Since our research focuses on how CAM is collected and represented, we focus on the first 4 layers.

### 2.1 TaskTracer

TaskTracer [Dragunov, 05] employs a key-value based model to enable knowledge workers to rapidly locate, discover, and reuse past processes they used to successfully complete tasks. TaskTracer separates user context collecting and processing phases via a Publisher-Subscriber architecture: the Publisher collects data about user activities and forwards this event to one or more Subscribers that can process the event data in a different way.

To collect user interaction with desktop resources, TaskTracer uses a COM plug-in attached to MS Office applications, a windows CBT 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), which stores it in a database and sends it to Subscriber applications for further processing. The Storage/Management layer is implemented as a relational database, while the SQL query language represents the query layer.

### 2.2 Swish

Another key-value model based approach is SWISH [Oliver, 06], a knowledge work support, which automatically detects the tasks that the user is involved in, by identifying which of the desktop windows are related to each other. Like TaskTracer, it constantly monitors desktop activities and processes this information in order to perform recommendations. Unlike TaskTracer, which is restricted to a set of predefined traced applications, SWISH monitors window events generated by any application on a Windows PC.

As collecting sensors, SWISH uses a hook into the Windows OS which listens for the events produced by every window on the system. The reception layer is represented by LogFeeder, an interface between sensors and the storage/management layer. LogFeeder listens for windows events, transfers them into an SQL database for later processing and replays them in real time to an external component for live processing.

### 2.3 Contextualized Attention Metadata

[Wolpers, 07] introduces the Contextualized Attention Metadata (CAM) schema (see Figure 1) and architecture to capture behavioural information of users within different applications. The main elements of the schema are the *group* element which incorporates user attention in all systems, the *feed* element which groups the attention of the user in one specific system, the *item* element which collects the attention given to one specific digital document, and the *event* element which represent the events occurring on documents (e.g. read, update, download, etc.).

The CAM architecture allows collecting attention metadata from any desktop or server side application, merges them into a single stream per user, and stores data in the CAM store. The Sensor layer of the current CAM implementation comprises various tools and add-ons to collect attention metadata: the AriadneFinder [AriadneFinder, 10] and the MACE project [Mace, 10] use integrated agents to collect CAM , whereas the CAMera framework [Scheffel, 09] uses add-ons for Thunderbird email-client, Skype chat-messenger, Firefox browser, MS Outlook, the file system, MS Power Point, MS Word and the Flash meeting system. The reception layer is represented by Simple Publishing Interface (SPI) [SPI, 08] for AriadneFinder and MACE system, and by a database connectivity API for CAMera. XML based repositories represent the storage layer for all three implementations. The Simple Query Interface (SQI) [SQI, 05] is used as the query layer for MACE and AriadneFinder, and SQL for CAMera.

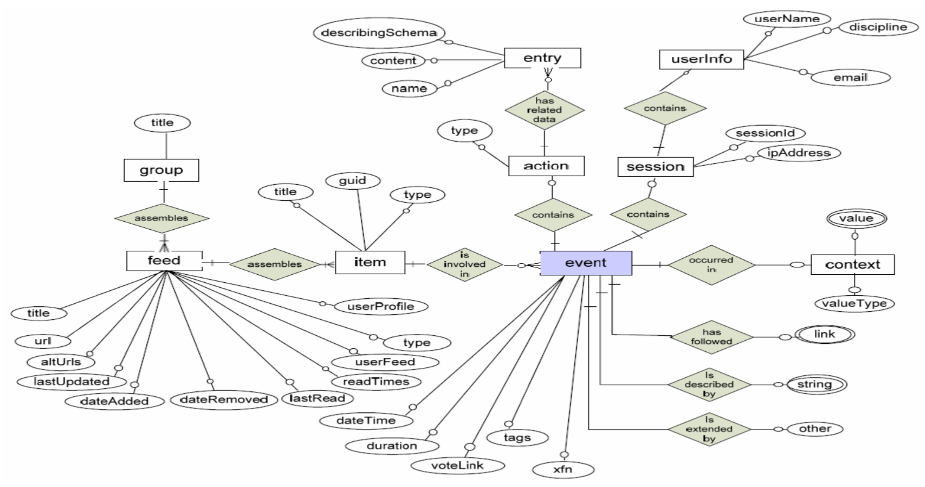


*Figure 1: The CAM schema elements [Wolpers, 07]*

## 2.4 Dyonipos

Dyonipos [Rath, 09] is a Personal Information Management (PIM) application that automatically identifies the work task of the user and then provides him with information from both personal and organizational environments. In Dyonipos, the context of the user is seen as a semantic pyramid implemented by a user interaction context ontology composed of five dimensions [Rath, 09]: (1) the *action* dimension which consists of concepts representing user actions, (2) the *resource* dimension that contains concepts for representing resources in the desktop computer, (3) the *user* dimension that integrates concepts about the user, (4) the *application* dimension, a hidden dimension represented by a property of the Event concept, and (5) the *information need* dimension, the context-aware pro-active information delivery part of the ontology.

Dyonipos system uses two types of sensors for capturing context data: sensors for mainstream applications (MS Word, PowerPoint, Excel, Internet Explorer, Microsoft Explorer, Firefox, Thunderbird and the Novell GroupWise email client) and sensors for the operating system (the file system, clipboard, network stream and generic Windows XP System Sensors). These sensors send usage data to the reception layer, which is represented by the Dyonipos Task Recognizer. The role of this component is to process and analyze the contextual data for storage in an RDF repository. An RDF query language is used for querying this repository.

## 2.5 Context Modeling Language

One graphical context modeling approach is Context Modelling Language (CML) [Henricksen, 06], an extension of Object-Role Modelling (ORM) [Halpin, 06]. It describes types of information, their classification (sensed, static, profiled or derived), relevant quality metadata, and dependencies amongst different types of information. Moreover, it supports a variety of constraints, both general (such as cardinality of relationships) and special-purpose (such as snapshot and lifetime constraints on historical fact types). In the case study presented in [Henricksen, 06], the reception and adaptation layers use the JDBC API and PostgreSQL for storage of facts, situations and preferences. The query layer is represented by the SQL query language.

## 2.6 WildCAT

WildCAT [David, 05] is an extensible Java framework to ease the creation of context-aware applications by providing a common interface to enable the integration and access to heterogeneous information. It contains a generic context model schema illustrated in Figure 2 and supporting different levels of extensions: the context is made of several domains that separate the different aspects of the context and allow each of these to use a custom implementation, while each domain is modeled as a tree of resources, each being described by attributes (simple key/value pairs). Finally, the context information can be accessed through both the pull and push modes.

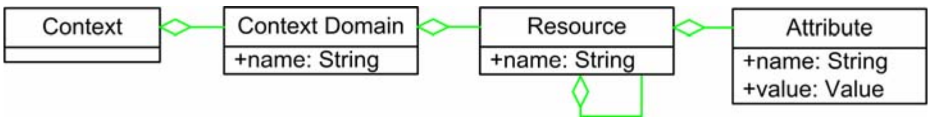


*Figure 2: WildCAT generic model [David, 05]*

## 2.7 Comparison criteria

Previous sections detailed various tracking approaches by focusing on their CAM representation and support architecture. This section aims at positioning these approaches regarding some criteria that frameworks collecting CAMs from different applications have to deal in order to federate the existing CAMs and to perform recommendations to users on their bases. These criteria are the followings:

**User profile**. [Kobsa, 99] makes a distinction between user data, usage data, and environment data: user data (or user profile) comprise the various characteristics of the users (interest, knowledge, goals, etc.), usage data are composed of data about users' activities on resources and systems, and environment data refer to data that are not related to the users themselves (e.g. data about applications, systems, resources, etc.). A prerequisite for developing personalized services able to provide the user with the right information, at the right time, through the right means, is to rely on user profiles representing users' information needs. Thus, it is important to represent the user profile (or user data) within the context model.

**Model flexibility**. Users usually interact with various tools and applications. To relate together users' interactions within different applications, as well as users between them, we need a flexible model which models context specific to each application, offers a unified view of observed data, and allows new CAM to be captured when new applications and resources emerge.

**Open framework**. The framework has to be open to permit the simple integration of new collecting sensors. A model can be application-domain bounded if it focuses on a single application or specific domain, or can be fully abstract if it can naturally deal with different domains or applications (e.g. it is possible to capture any kind of context with this model).

**Model expressiveness**. The model should be able to express the context of the user at different granularity levels (e.g. class hierarchy) and relationships between different subcomponents of the user context (e.g. composition, association relationships).

**Context sharing**. In order to allow multiple applications and services to be built upon the context information, CAM data have to be easily accessible.

**Context constraints**. The context model survey of [Bolchini, 07] concludes that the practical applicability and usability are often inversely proportional to the generality of the model: the more expressive and powerful, the less practical and usable. Thus, it is important to impose constraints that the contexts must satisfy for a given application, i.e. application oriented relationships between context components or predefined names of context metadata.

**Scalable framework**. As new context data is captured, and new applications are observed, the amount of available information grows, so scalable frameworks are needed in order to deal with a huge number of stored data.

Based on these criteria, the presented approaches are compared in Table 1. Due to their very simple structure, key-value models, TaskTracer and Swish are on the one hand neither flexible nor expressive, but implement context constraints (data type constraints) on the other hand. They do not take into account a detailed user profile and the context is enclosed within a relational database, thus preventing context sharing. Moreover, they are built upon specific agents, thus preventing openness.

| | Model | | | | | Architecture | | |
|---|---|---|---|---|---|---|---|---|
| | Type of formalism | User profile | Model flexibility | Model expressiveness | Context constraints | Context sharing | Framework openness | Framework scalability |
| TaskTracer | Key -value | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| Swish | Key -value | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| CAM | Mark-up | ✓ | + | + | ✗ | ✓ | +++ | ✓ |
| Dyonipos | Ontology | ✓ | ++ | +++ | ✓ | ✗ | ++ | ✗ |
| CML | Graphical | ✓ | ++ | +++ | ✓ | ✗ | + | ✗ |
| WildCAT | Obj. oriented | ✓ | ++ | ++ | ✗ | ✗ | + | ✗ |

Implemented feature: ✓ (yes), ✗ (no)
Feature level: + (low), ++ (medium), +++ (high)

*Table 1: Comparative table*

Within CAM, the user profile is defined by two elements. The first one, *userProfile*, is a reference to the profile of the user within the observed application. The second element, *session*, provides information about the session, such as IP address, username or email address. The model has difficulties in modeling context specific to different applications and resources due to predefined and fixed context metadata attributes suggested by the majority of schema elements (e.g. the *item* element is defined by three properties, and it is not possible to define additional attributes). Only elements characterized by "*" multiplicity (e.g. the *entry* element) allow defining new context metadata information; it is possible to define new elements, but one can create a "name" element while someone else can create a "surname" element to identify the same data. Therefore, elements that can be extended cannot be validated against the XML schema. Moreover, the absence of relationship and granularity level between context components prevent CAM to offer a very expressive model. Thanks to SQI web services it is possible to widely share the containing attention metadata. If new applications have to be traced the only thing to do is the implementation of the agents within the concerned applications. The structure of agents is very simple, comprising the code to call SPI web service plus the code to create an XML string comprising the users' context which is conform to the CAM schema. The architecture also has a high level of scalability, due to its Service Oriented Architecture.

In Dyonypos, the user profile is composed of two concepts: user and session. The user concept defines basic user information such as user name, password, first name and second name, whereas the session concept is used to track the time of user logins and the duration of a user session in the application. Compared to CAM, the model of Dyonipos can be easily extended with concepts, properties about new resources, user actions and relationships between resources on various granularity levels; it thus provides a high expressiveness and context validation. On the other hand, Dyonipos shares context data between workers of the same organization, so the context information is enclosed to the organizational level within a central server which contains the context data of all users. Due to its centralized architecture, scalability loss may appear in Dyonipos.

In CML, the user profile is not explicitly defined, but it can be modeled as facts. The addition of new facts and situations in the CML model is largely automated by a tool that inputs textual representations of fact types and situations and, based on a relational mapping, generates scripts that manipulate the relevant databases (the implementation of new sensing agents is still required). In this implementation [Henricksen, 06], the context is enclosed within a central relational database, thus preventing framework scalability. In addition, it is very difficult to share the user context, since a login and a password are required to access data stored into the database. CML facts and relations between facts imply a high level of context expressiveness and context validation.

In WildCAT, the user profile is not explicitly defined, but it can be modeled as context domain class. The model is highly flexible and extensible due to the free form text values of the Attribute class and its high abstraction level, but the relationships between context components are missing: there are no context constraints. The context information can be accessed by Java applications only, thus limiting interactions with this framework.

As a conclusion of our survey, we notice that there is no approach that entirely fulfils the whole set of criteria defined earlier. On the one hand, even if some approaches as Dyonipos, CML and WildCAT have a highly expressive model, they lack scalability and openness. On the other hand, even if CAM is scalable and highly supports context sharing, it lacks expressiveness. Therefore, there is a need for an approach which is as much as expressive and flexible as Dyonypos, and at the same time, open, scalable and promoting context sharing as CAM does.

## 3 Modeling and managing tracking data

In the systems and networks area, knowing the state of the operating systems, applications or networks is a major concern since the 90's. Thus, a standard emerged to supervise any network or computer system. In fact, 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.

To represent managed data in a common way, the DMTF standard adopted a Common Information Model (CIM) [CIM, 98], a meta-model composed of three main schema representing information to supervise: (1) the core models capture notions that are applicable to all management areas, (2) the common models extend the core models and contain information models that represent notions that are common to particular management areas but independent of any particular technology, and (3) the extension schema represents technology-specific extensions of the common models.

To support this meta-model, the DMTF introduces a distributed architecture called Web-Based Enterprise Management (WBEM) [WBEM, 99]. The main components of the WBEM architecture are the manager, which is responsible for processing and storing management information, the providers, which are pieces of software that communicate with the managed resources in order to access data and to forward it to the manager, and the client applications, which interact with the manager in order to retrieve or set management information.

CIM represents the basis for our CAM models detailed in the next section. Here, we focus on our global architecture based on the WBEM standard but applied to our TEL environment (see Figure 3). The architecture is divided into three parts:

– The first part represents user tools from which attention metadata are collected. These *direct access sensors* correspond to the *data gathering layer* of the six layered architecture of Henricksen (see Section 2).
– The tracking environment represents both the *storage/management* and the *query layers*. Indeed, the tracking repository is responsible for storing attention information, whereas the tracking manager is able to manipulate CAMs stored into the repository using the CIM Query Language (CQL).
– The intermediate layer (*reception layer*) bridges the gap between the user contexts on the tracking environment. Thus, learning (or non-learning) tools are able to easily provide and/or retrieve CAM stored into the repository.

We extended the core CIM models in order to cover the particular area of TEL. The resulting models have a high abstraction level, containing a set of classes, associations and properties that provide a basic vocabulary for describing more specific tracking objectives.

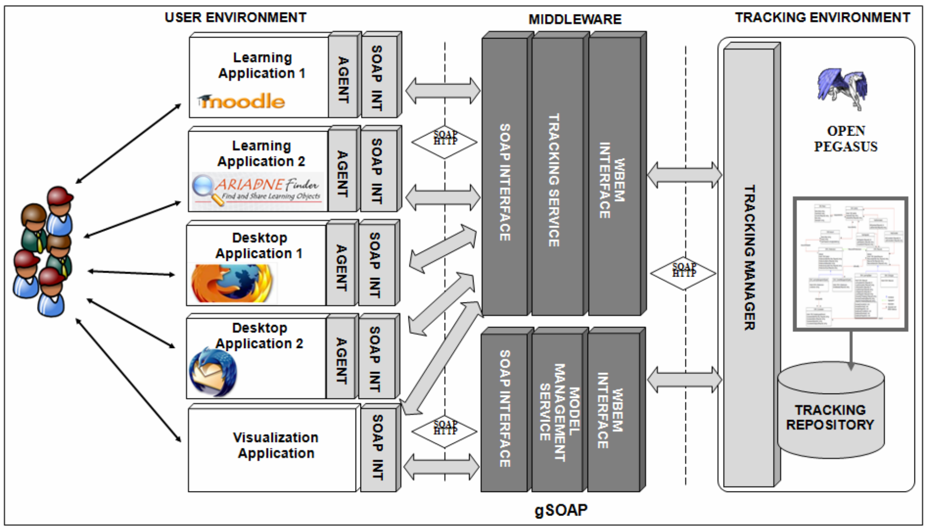


Figure 3: The global architecture

## 3.1 Generic CAM models

The tracking model must be able to take into account any attention metadata related to users, tools, resources and activities. Therefore, we defined two generic models composed of two sub models: the model TEL_Environment (see Figure 4) focuses on learning systems and resources, whereas the model TEL_Activity (see Figure 5) aims at describing interactions of users with these systems and resources. Each of these models presents a high abstraction level, and offers the opportunity of defining specific models according to specific objectives. This section only exposes the higher models, since the specific models are related to the implementation presented in section 4. Moreover, the user profile is represented on Figure 4 as the class CIM_Identity; it is precisely detailed in [Ramanda, 09], and includes the IMS-LIP standard [IMS, 01] together with some additional information.

The main classes of the TEL_Environment model are TEL_ApplicationSystem and TEL_Resource; they respectively model any systems and resources. Since these systems/resources can be composed of others systems/resources, we introduced two composition relationships (respectively TEL_SystemComponent and TEL_ResourceComponent). In addition, another composition (TEL_SystemResourceComponent) expresses the fact that a system hosts resources. Finally, in order to link a user with a system or resource, we respectively designed the associations TEL_IdentityOnSystem and TEL_IdentityOnResource.

To identify an activity processed by a user on a resource, we introduced the association TEL_DependencyResourceActivity (see Figure 5). Activities operated by users are represented through the root class TEL_ResourceActivity, and compositions between activities are expressed by the composition

TEL_ResourceActivityComponent. Figure 5 only depicts activities related to resources; the generic model dedicated to activities operated on systems follows a similar reasoning.

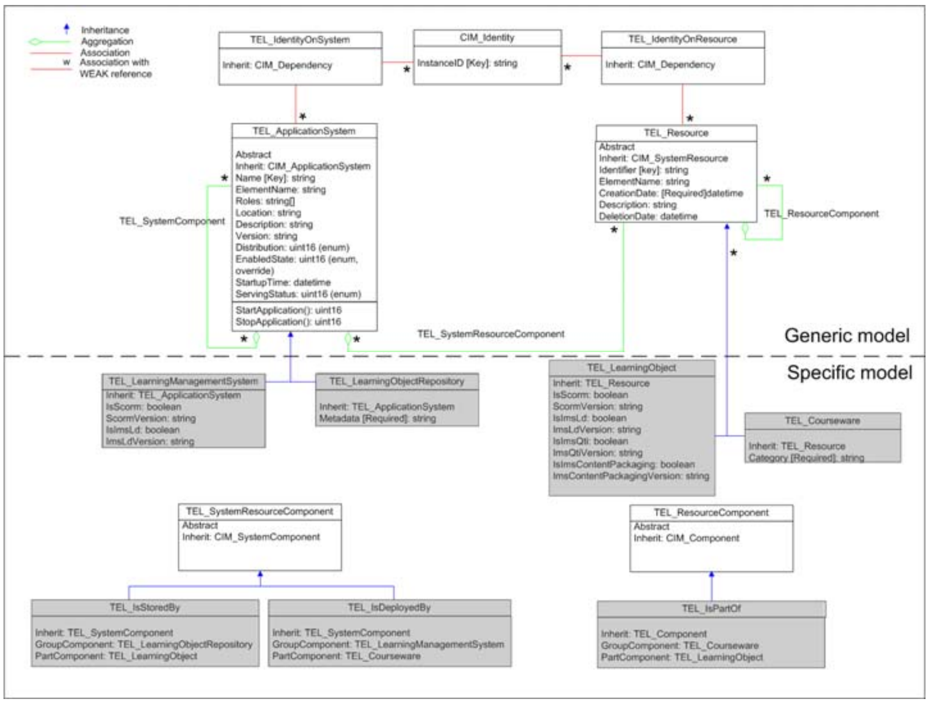


*Figure 4: The TEL_Environment model*

As already mentioned in section 2.7, models whose aim is to be completely general to support the context modeling problem as a whole for any possible application, often fail to be effective: the more expressive and powerful, the less practical and usable. Our model is both *expressive* due to context granularity levels and relationships, and *practically applicable* because it has a well defined focus and tries to support only a specific context subproblem, thus dealing with context referring to systems, resources, users, activities, and not to any kind of context (e.g. the WildCATs' context superclass tries to model any kind of context, contrary, we define the resource, system and activity classes and not the context class). A detailed comparison of our approach to existing models is discussed in Section 5.

Our model is based on existing CIM classes. Therefore, we can make use of information described by our model, but also benefit from other useful information already specified within existing CIM models such as information about operating system, installed peripheral devices or screen resolution; this information enriches the description of the user context. Since data are enclosed within a WBEM repository, we introduced a middleware layer in order to share the existing data. The middleware contains two services: the model management service that interacts with the classes of the model, and the tracking service that interacts with the class instances.

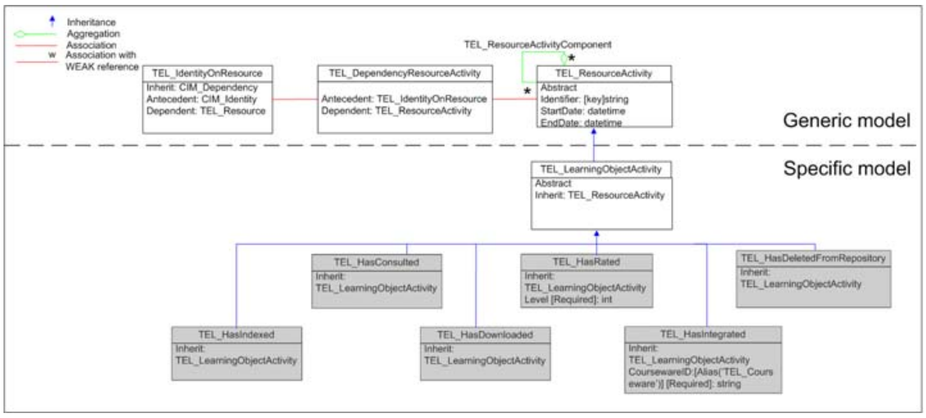

Inheritance
Aggregation
Association
w Association with WEAK reference
TEL_ResourceActivityComponent
TEL_IdentityOnResource
Inherit: CIM_Dependency
Antecedent: CIM_Identity
Dependent: TEL_Resource
TEL_DependencyResourceActivity
Antecedent: TEL_IdentityOnResource
Dependent: TEL_ResourceActivity
TEL_ResourceActivity
Abstract
Identifier: [key]string
StartDate: datetime
EndDate: datetime
Generic model
Specific model
TEL_LearningObjectActivity
Abstract
Inherit: TEL_ResourceActivity
TEL_HasConsulted
Inherit: TEL_LearningObjectActivity
TEL_HasRated
Inherit: TEL_LearningObjectActivity
Level [Required]: int
TEL_HasDeletedFromRepository
Inherit: TEL_LearningObjectActivity
TEL_HasIndexed
Inherit: TEL_LearningObjectActivity
TEL_HasDownloaded
Inherit: TEL_LearningObjectActivity
TEL_HasIntegrated
Inherit: TEL_LearningObjectActivity
CoursewareID:[Alias('TEL_Courseware')] [Required]: string

*Figure 5: The TEL_Activity model for resources*

CIM is described by Managed Object Format (MOF) [CIM, 98], a language to describe object-oriented classes and instance definitions in textual form, with the goals of human readability and parsing by a compiler. Using the MOF compiler, it is easy to integrate our models to any implementation of the WBEM standard.

## 3.2 The CAM services

### 3.2.1 The model management service

When specific data must be tracked, there is a need for specializing the generic models. Instead of manually compiling classes and properties matching with new attention information to supervise, we introduced the model management service (see Figure 3) that contains a set of methods (see Table 2) to extend the classes of the generic models. Some of the methods process on classes, while others work on attributes. The whole set of methods includes common data transactions required to manage data: insertion of a new class or a new attribute into an existing class, update of an existing class or attribute, and deletion of an existing class or attribute. However, some restrictions apply: one can only define new classes inheriting from the generic environment or activity models, and one cannot modify the specifications of the generic models.

This service also offers the opportunity to retrieve the global tracking model. Basically, it returns as an XML string the environment model, the activity model, and the user model. One can thus discover traces that are made available.

The model management service ensures extensibility of the model, and thus allows the specification of any attention data. The service presented in the next section gathers CAMs resulting from user activities, and inserts the matching instances into the tracking repository.

| *Method name* | *Parameters* | | *Retur n type* | *Fault* |
|---|---|---|---|---|
| | *Name* | *Type* | | |
| **addClass** | class | String | Void | STRING_NOT_VALID<br>CIM_ERR_INVALID_CLASS |
| **delClass** | className | String | Void | CIM_ERR_INVALID_CLASS |
| **updClass** | class | String | Void | STRING_NOT_VALID<br>CIM_ERR_INVALID_CLASS |
| **addProperty** | property | String | Void | XML_STRING_NOT_VALID<br>CIM_ERR_INVALID_CLASS |
| **delProperty** | property | String | Void | STRING_NOT_VALID<br>CIM_ERR_INVALID_CLASS<br>CIM_ERR_INVALID_PARAM |
| **updProperty** | property | String | Void | STRING_NOT_VALID<br>CIM_ERR_INVALID_CLASS<br>CIM_ERR_INVALID_PARAM |
| **getModel** | | | String | |

*Table 2: Methods of the model management service*

### 3.2.2 The tracking service

The tracking service plays two distinct roles through two methods (see Table 3): the first one transforms the XML traces received from collecting agents into CIM instances and inserts them into the CAM repository; the second one retrieves the existing CAMs by querying the tracking manager.

| *Method name* | *Parameters* | | *Return type* | *Fault* |
|---|---|---|---|---|
| | *Name* | *Type* | | |
| **publishTrace** | trace | String | Void | TRACE_NOT_VALID |
| **queryTrace** | query | String | String | INVALID_QUERY |
| | resultFormat | String | | INVALID_FORMAT |

*Table 3: Methods of the tracking service*

The publishTrace method is able to receive traces produced by any web-based system, and to build or modify the matching instances defined within the CAM model. In order to accomplish these tasks, the service has to take into account both the generic and the specific models. Therefore, this method queries the tracking manager to get the class definitions and then builds the matching XML schema each time the model is modified. Through this mechanism, this method is able to process any trace compliant with the global model.

The queryTrace method explores the CAM repository in order to get a unified view of all traces. Treatments associated to this method consist in querying the tracking manager to retrieve CIM and TEL instances of the model using the CIM Query Language [CQL, 04] elaborated by the DMTF. Figure 6 shows a CQL query example to retrieve the object paths of all learning objects that user Joe interacted with. The service is dynamic in the way that the query applies not only to the specified classes but also to their child classes, so that even if new classes or attributes are defined into the model, they will be retrieved and returned to computer systems or users. In addition, this method is able to return the trace according to a specific format specified as a parameter.

```
SELECT  TEL_IdentityOnResource.Dependent
FROM    TEL_IdentityOnResource, TEL_DependencyResourceActivity,
        TEL_LearningObjectActivity
WHERE   TEL_IdentityOnResource.Antecedent=”CIM_Identity.InstanceID=/”Joe/””
        AND TEL_DependencyResourceActivity.Antecedent=
              OBJECTPATH(TEL_IdentityOnResource)
        AND TEL_DependencyResourceActivity.Dependent=
              OBJECTPATH(TEL_ LearningObjectActivity)
```

*Figure 6: CQL Query sample*

## 4 Implementation

To validate our approach, a framework has been implemented to track usage of learning objects within two heterogeneous systems: the ARIADNE Learning Object Repository (LOR) [ARN, 96] and MOODLE [Moodle, 02], one of the most popular LMSs. The next section details the specific models that integrate attention data related to these systems and resources, then introduces the software deployed within our implementation, exposes a use case, and finally presents our experimentations.

### 4.1 The models specific to LOR, LMS and LO

The models specific to learning objects, LOR and LMS are described by the gray classes of the Figure 4. Two types of systems have been defined (TEL_LearningManagementSystem and TEL_LearningObjectRepository) and inherit from the root class to represent an application, whereas two types of resources (TEL_LearningObject and TEL_Courseware) specialize the higher class to model a resource. Moreover, some compositions illustrated at the bottom of Figure 4 express the fact that (1) a learning object is stored into a LOR (class TEL_IsStoredBy), (2) a courseware is deployed on a LMS (class TEL_IsDeployedBy), and (3) a learning object may be part of another learning object or courseware (class TEL_IsPartOf).

Activities that can be processed on learning objects are depicted in Figure 5 (the gray classes); we identified the indexation, consultation, download, rating and deletion operations. The class TEL_HasIntegrated translates the integration of a learning object within a courseware.

### 4.2 Open source software of our architecture

The implemented architecture is composed of the following components (Figure 3):

– The learning systems to supervize: the tool interacting with the ARIADNE LOR (called Finder) and MOODLE. These systems embed an agent responsible for collecting attention information when an activity is operated by a user.

– The tracking framework integrates OpenPegasus [OPE, 02], a C++ open source implementation of the WBEM standards. The repository thus contains classes of the generic and specific models.
– The services of the middleware layer are developed using gSOAP [GSP, 02] tools for development of SOAP/XML web services.

The next section details interactions between the above entities during the collection of attention information resulting from the consultation of a learning object by a user connected to a MOODLE server.

### 4.3 Use case: consultation of a learning object from Moodle

Figure 7 represents the UML sequence diagram illustrating exchanges and treatments required to produce and store a trace translating the consultation of a learning object by a student via MOODLE. When a learner consults a document on MOODLE (1), the integrated agent creates the matching XML trace (which contains information about the user, the learning object, the learning system and the activity) (2) and sends a request to the publishTrace method of the tracking service (3). The tracking service then validates the XML trace against the XML schema (4) and builds the matching instances of the models (5). These lasts are finally sent to the tracking manager (6) and stored into the CAM repository (7).

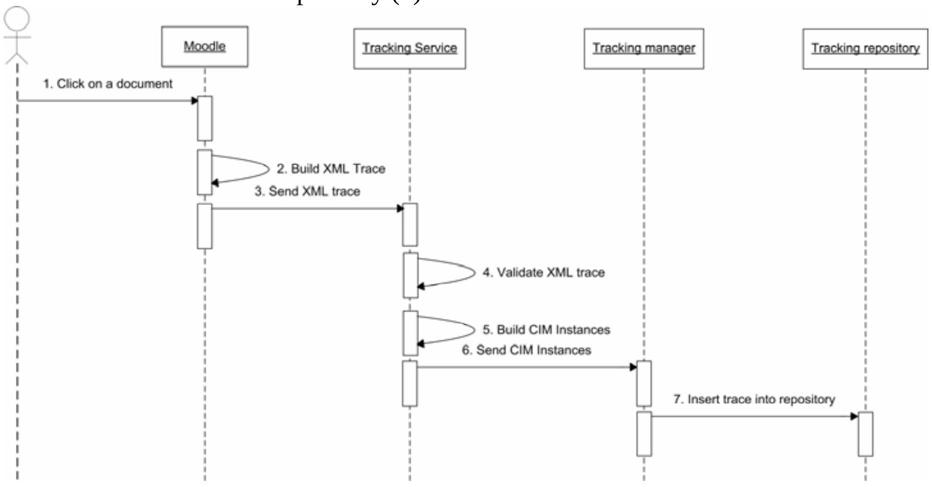
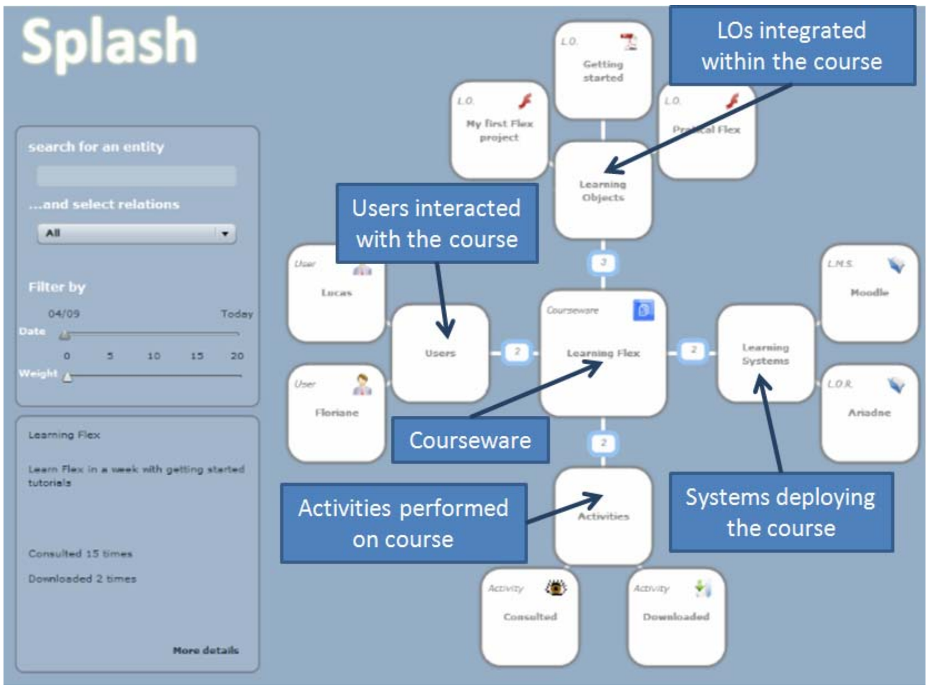


*Figure 7: Collecting a trace*

### 4.4 Current implementation

Our current implementation tracks the user context within the Ariadne Finder and Moodle. In the Ariadne Finder, the following user interactions are collected: the keywords used to search for LOs, LO metadata consultation, LO content download and indexation. In Moodle, user activities performed on both courseware and learning objects are gathered: the integration of a LO within a courseware, the consultation of a courseware/LO, the download, consultation and evaluation of LOs.

The collection of CAM from these applications started recently, and till now we have 1500 activities (instances of the TEL_ResourceActivity subclasses) performed on 1000 resources (instances of the TEL_Resource subclasses). A CAM visualization client application called SPLASH (Secured and Personal Learning dASHboard) is under development to visualize information stored into the tracking repository. This tool queries the tracking service to exploit relationships between model components and to provide end-users with an easy-to-use interface. In Figure 8, the courseware *Learn Flex* is hosted on a Moodle server and comprises three learning objects; the course has been indexed into the Ariadne repository, and two users have consulted and downloaded the courseware.



Learning Flex

Learn Flex in a week with getting started tutorials

Consulted 15 times

Downloaded 2 times

*Figure 8: SPLASH*

## 5 Discussion

The previous section gave an overview of the current status of our implementation of our framework. In this section, we discuss our approach compared to existing approaches and to the entire set of requirements defined in section 2.7.

The *user profile* is a sub-model of the general context model. Like environment and activity models, the profile model is generic and allows defining user profiles specific to different applications (see [Ramanda, 09] for more details).

Like Dyonipos, CML and WildCAT, our model is *highly flexible*, thus allowing for modeling user context specific to any learning or non-learning application or tool. This is possible by specializing the general purpose classes and their related association and composition classes to model users' interaction context to different granularity levels. Modeling context at different granularity levels and the presence of relationships between context components makes our model a *highly expressive* one.

Like CAM, our framework is *highly open*. When a new application has to be observed, only two steps are required: to describe the application and its associated resources and activities in terms of CAMs using the model management service, and to integrate an agent into the target application. Since the tracking service is XML-oriented, agents can be built in any programming language.

Concerning context constraints, we introduced a fixed structure of the generic model, specific relationships constraints between model components as well as predefined class attributes names (like CML and Dyonipos and contrary to free value properties of CAM and WildCAT). As an example, we specialized the generic relation TEL_SystemResourceComponet to specific applications: a LO is stored by a LOR (class TEL_IsStoredBy). If we would not have specialized it, there was no way to know that the LO is stored within a LOR and not within a LMS, thus introducing altered user context. Moreover, the XML string of the publish service is validated against an XSD Schema which represents an up-to-date structure of the CAM model.

The DMTF proposes a manager to manager (M2M) communication protocol to make two managers communicate with each other in order to exchange management data. Using this protocol we propose a hierarchical structure of managers to distribute charge and to deal with the scalability issue.

Like CAMs' SQI interface, our approach also uses a service oriented interface to share context data. The CQL standard language is used to explore the CAM repository in order to offer a unified view of all traces.

## 6 Conclusion and future works

A required step to provide learners with personalized learning processes consists in collecting, storing and facilitating access to data that characterize users, applications and interactions of users with these applications. The paper identifies a set of criteria that we consider of the most importance when applications come to deal with context-aware features, and investigates several existing initiatives and projects regarding these criteria. This study led us to propose a generic approach able to collect and share attention information produced by heterogeneous systems, and integrating the considered criteria. Our framework is based on the DMTF standardized architecture which is natively implemented in most nowadays operating systems to facilitate local and remote management of hosted systems and applications. Therefore, our framework can easily be deployed in any WBEM compliant implementation.

The user context modeling extends the native DMTF models to represent attention information related to personalized learning, and integrates a generic and extensible user profile. The resulting models are highly expressive and flexible, thanks to (1) various context granularity levels and relationships between these context components, and (2) the extensible character of our approach. The support architecture is characterized by a distribution of the tracking components that ensures scalability: entities responsible for collecting user contexts are embedded into targets applications, and data are stored into a dedicated repository. To make our framework open and to encourage context sharing, we introduced an intermediate layer composed of services for facilitating extension and specialization of the generic models and for offering wide access to user contexts. Finally, integrity of contexts models and data are ensured by constraints defined in the models themselves.

When the user context models evolve, components responsible for collecting data are not operational because information they produce does not conform to the model anymore. To automatically update these components, common update mechanisms implemented within nowadays operating systems or applications could be reused: components send scheduled requests to a server in order to compare the current and the latest versions. However, to optimize this process and to have an overview of the supervised systems and applications, we plan to set up a framework for management of collecting entities that are deployed over systems and applications. The most natural way to do this is to reuse our approach: the model already exists (it is possible to express the fact that a system or resource is integrated in or deployed by another system or resource), the only thing to do is to specify activities specific to these components (installation, configuration, update, etc.).

Concerning privacy of the user, we plan to use an encryption algorithm (hash function) implemented within the tracking service which codifies every incoming CAM and stores them within repository, together with an OpenID provider which offers, once an user is authenticated, his personal information used for encryption. Within SPLASH, an authenticated user can visualize his CAMs together with others anonymous CAMs. Another short-term perspective of this research consists in collecting information describing devices available to the learner. Indeed, more and more devices such as smart phones, touchpad or touch screens are today available on the market and require an automatic adaptation of the learning tools or resources. Since devices are natively specified within the DMTF models, the only thing to do to take into account devices in our framework is to define relationships between the native models and those presented in this paper.

In addition, we envisage improving SPLASH with useful statistical and personalization features for teachers and learners. For a given class, each student can visualize his and his colleagues' progress in acquiring a competence or a skill. The interface will recommend him tools, persons, resources adapted to his profile, difficulties and context. Thanks to the tracking service, SPLASH will be able to retrieve learners with the same preferences, thus encouraging social learning.

At the moment, our tracking framework is able to gather CAM from different applications and to store them in a uniform way. The next step of this research consists in exploiting attention information stored into the repository(ies) in order to provide learners with personalized learning sessions, tools or resources according to their contexts. We plan to use the stored CAMs to automatically detect the task of a user by means of machine learning techniques and to provide him with resources according to his task. The recommendation system will be developed as an independent service which interacts with the tracking manager to retrieve CAM, and with the user interface to provide the recommended material.

**Acknowledgements**

Katrien Verbert is a Postdoctoral Fellow of the Research Foundation - Flanders (FWO).

## References

[AriadneFinder, 10] AriadneFinder, 2010, http://ariadne.cs.kuleuven.be/AriadneFinder

[ARN, 96] The Ariadne Foundation, 1996, http://www.ariadne-eu.org

[Baldauf, 07] Baldauf, M.: A survey on context-aware systems, In Int. J. Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, 2007

[Blackboard, 97] Blackboard, 1997, http://www.blackboard.com/

[Bolchini, 07] Bolchini, C., Curino, A., C., Quintareli, E., Schreiber, A., F., Tanca, L.: A Data-oriented Survey of Context Models, In SIGMOD Record, Vol. 36, No.4, pp.19-26., 2007

[Chen, 04] Chen, H., Perich, F., Finin, T., Joshi, A.: SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications, In Int. Conf. On Mobile and Ubiquitous Systems: Networking and Services, August 2004

[CIM, 98] Systems Management: Common Information Model (1998). Open Group Technical Standard C804, DMTF, ISBN: 1-85912-255-8, 1998.

[CQL, 04] DMTF CIM Query Language Specification, 2004, http://www.dmtf.org/standards/documents/WBEM/DSP0202.pdf

[David, 05] David, P-C., Ledoux, T.: WildCAT: a generic framework for context-aware applications, In MPAC '05, Grenoble, France, November 28 – December 2, 2005

[Dey, 01] Dey, A.K.: Understanding and using context, In Personal and Ubiquitous Computing Journal 1(5), 4-7

[Dragunov, 05] Dragunov, A., N., Dietterich, T., G., Johnsrude, K., McLaughlin M., Li, L., Herlocker J., L.: TaskTracer: a desktop environment to support multi-tasking knowledge workers, In Proc. IUI '05 (2005), 75-82, 2005

[Groza, 07] Groza, T., Handschuh, S., Möller, K., Grimnes, G., Sauermann, L., Minack, E., Jazayeri, M., Mesnage, C., Reif, G., Gudjónsdóttir, R.: The NEPOMUK Project – On the Way to the Social Semantic Desktop. Proceedings of the Third International Conference on Semantic Technologies (I-SEMANTICS 2007), Graz, Austria, 2007.

[GSP, 02] Engelen, R., A., Gallivan, K.: The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks, In Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), 128-135, Berlin, Germany, May 21-24, 2002

[Halpin, 06] Halpin, T.: Object-Role Modeling (ORM/NIAM), In Handbook on Architectures of Information Systems, pages 81-103, Springer Berlin Heidelberg, 2006

[Henricksen, 06] Henricksen, K., Iadulska, J.: Developing context-aware pervasive computing applications: Models and approach, In Pervasive and Mobile Computing Vol. 2, Issue 1, February 2006, 37-64, 2006

[IMS, 01] IMS Learner Information Package Best Practice & Implementation Guide, 2001, http://www.imsglobal.org/profiles/lipbest01.html

[Kobsa, 99] Kobsa, A., Koenemann, J. and Pohl, W.: Personalized hypermedia presentation techniques for improving online customer relationships. Technical report No. 66 GMD, German National Research Center for Information Technology, St. Augustin, Germany.

[Lindstaedt, 06] Lindstaedt, S., Mayer, H.: A Storyboard of the APOSDLE Vision. European Conference on Technology Enhanced Learning (EC-TEL'2006), Crete, Greece, 2006.

[MACE, 10] Metadata for Architectural Contents in Europe (MACE), 2010, http://portal.mace-project.eu/

[Moodle, 02] Moodle, 2002, http://moodle.org/

[Oliver, 06] Olivier, N., Smith, G., Thakkar, C., Surendran, A., C.: SWISH: semantic analysis of window titles and switching history, In Proceedings of the 11th international Conference on intelligent User interfaces (Sydney, Australia, January 29 - February 01, 2006). IUI '06. ACM, New York, NY, 194-201

[OPE, 02] Open Pegasus, 2002, http://www.openpegasus.org/

[Ramanda, 09] Ramandalahy, T., Vidal, P., Broisin, J.: Opening Learner Profiles across Heterogeneous Applications, In IEEE International Conference on Advanced Learning Technologies (ICALT 2009), Riga, Latonia, 14/07/2009-18/07/2009.

[Rath, 09] Rath, S., A., Devaurs, D., Lindstaedt, N., S.: UICO: An Ontology-Based User Interaction Context Model for Automatic Task Detection on the Computer Desktop, In Proceedings of the 1st Workshop on Context, Information and Ontologies, June 1, Heraklion, Greece, 2009

[Sauermann, 06] Sauermann, L., Grimnes, G., A., Kiesel, M., Fluit, C., Maus, H., Heim, D., Nadeem, D., Horak, B., Dengel, A.: Semantic Desktop 2.0: The Gnowsis Experience. 5th International Semantic Web Conference, Athens, GA, USA, November 5-9, 2006.

[Scheffel, 09] Scheffel, M., Friedrich, M., Jahn, M., Kirschenmann, U., Niemann, K., Schmitz H-C., Wolpers, M.: Self-monitoring for Computer Users, In Exploitation of Usage of Attention Metadata Workshop (EUAM ’09), Lubeck, Germany, 2009

[Schmidt, 07] Schmidt, A.: Impact of Context-Awareness on the Architecture of Learning Support Systems, Architecture Solutions for E-Learning Systems, Idea-Group Publishing, 2007

[Shen, 05] Shen, X., Tan, B., Zhai, C.: Context-sensitive information retrieval using implicit feedback, In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, Salvador, Brazil, 2005

[SPI, 08] Ternier, S., Massart, D., Van Assche, F., Smith, N., Simon, B., Duval, E.: A Simple Publishing Interface For Learning Object Repositories, In Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications, 1840-1845, Chesapeake, VA: AACE, 2008

[SQI, 05] Massart, D., Van Assche, F., Ternier, S., Duval, E., Brantner, S., Olmedilla, D., Miklós, Z.: A Simple Query Interface for Interoperable Learning Repositories, In Proc. of the 1st Workshop on Interoperability of Web-based Educational Systems, May 10, 11-18, 2005.

[Strang, 04] Strang, T., Linnhoff-Popine, C.: A context modelling survey, In Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp ’04, Nottingham, England, 2004

[Sugiyama, 04] Sugiyama, K., Hatano, K., Yoshikawa, M.: Adaptive web search based on user profile constructed without any effort from users, In Proceedings of the 13th international Conference on World Wide Web (New York, NY, USA, May 17 - 20, 2004). WWW '04. ACM, New York, NY, 675-684., 2004.

[WBEM, 99] Web Based Enterprise Management, 1999, http://www.dmtf.org/standards/wbem/

[Wolpers, 07] Wolpers, M., Najjar, J., Verbert, K., Duval, E.: Tracking Actual Usage: the Attention Metadata Approach, In Educational Technology & Society, 10 (3), 106-121, 2007