

GRAPPLE

D7.1a Version: 1.0

Initial specification of the operational infrastructure

Document Type	Deliverable
Editor(s):	Lucia Oneto (GILABS)
Author(s):	Lucia Oneto (GILABS), Marco Poggi (GILABS), Michele Dicerio (GILABS) Dominikus Heckmann (DFKI), Eelco Herder (LUH), Kees van der Sluijs (TUE), Luca Mazzola (USI), Kai Höver (IMC), Ian Boston (UCAM), Frederic Minne (UCL)
Internal Reviewers	Cord Hockenmeyer (UniGraz), Vincent Wade (TCD)
Work Package:	WP7
Due Date:	31-1-2009
Version:	1.0
Version Date:	28-01-2008
Total number of pages:	43

Abstract: This document presents the initial specifications of the interfaces to be designed and developed in the GRAPPLE Operational Infrastructure, where all the different components of the system will be integrated (ALE, LMS, authoring tools, user interfaces, etc).

Keyword list: Learning Management System, Adaptive Learning Environment

Disclaimer:

All information included in this document is subject to change without notice.

The Members of the GRAPPLE Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the GRAPPLE Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Summary

In order to obtain a final system that is actually able to provide effective and efficient adaptive courses, it is necessary to identify the interfaces to be designed and developed in the GRAPPLE Operational Infrastructure, where all the different components of the system will be integrated (ALE, LMS, authoring tools, user interfaces, etc).

This document presents the initial specifications of the GRAPPLE Operational Infrastructure that includes the two components available for the first prototype: the existing LMSs present in the consortium and the first available version of the GRAPPLE core engine.

Authors

Person	Email	Partner code
Lucia Oneto	l.oneto@giuntilabs.com	GILABS
Marco Poggi	m.poggi@giuntilabs.com	GILABS
Michele Dicerio	m.dicerio@giuntilabs.com	GILABS
Dominikus Heckmann	heckmann@dfki.de	DFKI
Eelco Herder	herder@l3s.de	LUH
Kees van der Sluijs	k.a.m.sluijs@tue.nl	TUe
Luca Mazzola	luca.mazzola@lu.unisi.ch	USI
Kai Höver	Kai.Hoever@im-c.de	IMC
Ian Boston	ian@caret.cam.ac.uk	UCAM
Frederic Minne	frederic.minne@uclouvain.be	UCL

Table of Contents

SUMMARY 2

AUTHORS 3

TABLE OF CONTENTS 4

TABLES AND FIGURES..... 6

LIST OF ACRONYMS AND ABBREVIATIONS 7

1 TASK AND DELIVERABLE DESCRIPTION..... 9

2 INTRODUCTION 10

3 ARCHITECTURAL APPROACH 11

4 GRAPPLE OPERATIONAL INFRASTRUCTURE OVERVIEW 12

4.1 Scenarios and Use Cases 12

 4.1.1 Scenario 1 - Learner 12

 4.1.2 Scenario 2 - Learner 13

 4.1.3 Scenario 3 - Learner 21

 4.1.4 Scenario 4 - Author..... 24

4.2 Overall Architecture 26

 4.2.1 ALE components 26

 4.2.2 LMS 26

5 GRAPPLE KEY COMPONENTS – FIRST PROTOTYPE 32

5.1 GRAPPLE Core Engine - GALE (WP1)..... 32

 5.1.1 Functionality..... 32

 5.1.2 The GALE UM Service 33

6 APPENDIX 35

6.1 CLAROLINE 35

 6.1.1 Description of the LMS 35

 6.1.2 Integration with GRAPPLE ALE 35

 6.1.3 Limitations and Risks..... 35

6.2 MOODLE 36

 6.2.1 Description of the LMS 36

 6.2.2 Integration with GRAPPLE ALE 36

 6.2.3 Limitations and Risks..... 36



- 6.3 SAKAI 37**
 - 6.3.1 Description of the LMS 37
 - 6.3.2 Integration with Grapple ALE..... 37
 - 6.3.3 Limitations and Risks..... 37

- 6.4 IMC CLIX 37**
 - 6.4.1 Description of the LMS 37
 - 6.4.2 Integration with GRAPPLE ALE 38
 - 6.4.3 Limitations and risks 39

- 6.5 learn eXact 39**
 - 6.5.1 Description of the LMS 39
 - 6.5.2 Integration with GRAPPLE ALE 42
 - 6.5.3 Limitations and risks 42

- REFERENCES 43**

Tables and Figures

List of Figures

Figure 1: Scenario 1 – Use case diagram.	12
Figure 2: Scenario 1 – Activity diagram.....	13
Figure 3: Scenario 2 – Use case diagram.	15
Figure 4: Scenario 2 – Activity diagram.....	16
Figure 5: Scenario 2 –Activity diagram – first step.....	17
Figure 6: Scenario 2 – Activity diagram – second step.....	18
Figure 7: Scenario 2 – Activity diagram – third step.	19
Figure 8: Scenario 2 – Activity diagram – final step.....	20
Figure 9: Scenario 3 – Use case diagram.	21
Figure 10: Scenario 3 – Activity diagram.....	22
Figure 11: Scenario 3 – Activity diagram – third step.	23
Figure 12: Scenario 4 – Use case diagram.	25
Figure 13: LMS – Use case diagram.....	27
Figure 14: Learner scenarios – first prototype.	28
Figure 15: Author scenario – first prototype.....	29
Figure 16: Shibboleth & GRAPPLE.....	30
Figure 17: GRAPPLE first prototype architecture.	32
Figure 18: Detailed view on the learning management components of CLIX.....	38
Figure 19: learn eXact system.....	41

List of Acronyms and Abbreviations

ADL	Advanced Distributed Learning
AICC	Aviation Industry CBT (Computer-Based Training) Committee
ALE	Adaptive Learning Environment
API	Application Programming Interface
AS	Authentication Service
AXIS	Apache eXtensible Interaction System
CAM	Conceptual Adaptation Model
CAS	Central Authentication System
CMS	Content Management System
CRT	Concept Relationship Type
CSV	Comma Separated Values
DM	Domain Model
EB	Event Bus
ES	Event Service
GRAPPLE	Generic Responsive Adaptive Personalized Learning Environment
IMS	Instructional Management System
IMS CP	IMS Content Packaging
IMS LD	IMS Learning Design
IMS QTI	IMS Question and Test Interoperability
IMS SS	IMS Simple Sequencing
JMS	Java Message Service
LCMS	Learning Content Management System
LDAP	Lightweight Directory Access Protocol
LMS	Learning Management System
PHP	PHP Hypertext Preprocessor
SAML	Security Assertion Markup Language
SCORM	Shareable Content Object Reference Model
SOAP	Simple Object Access Protocol
SPIP	Système de Publication pour l'Internet Participatif
SSO	Single Sign On
UM	User Model
WAYF	Where Are You From
WP	Work Package
WSDD	Web Service Deployment Descriptor
WSDL	Web Services Description Language
VLE	Virtual Learning Environment



VRE	Virtual Research Environment
XML	eXtensible Markup Language

1 Task and Deliverable Description

T 7.1 Operational infrastructure specification and design (GILABS, ATOS, TUe, LUH, IMC)

This task will result in the design of the overall GRAPPLE Operational Infrastructure where all the different components of the system will be integrated (Adaptive Learning Environments, Learning Management Systems, User Model Framework, authoring tools, user interfaces, etc.). The main activities will include an analysis of the components selected/developed by other WPs, the identification of the communication channels amongst them and the design of suitable interfaces. The communication will be realized through web service architecture. Available commercial and open source LMSs will be considered and analyzed (Claroline, Sakai, Moodle, but also others, e.g. Learn eXact and IMC CLIX) in order to understand and model the underlying requirements for interoperability in existing learning environments.

D7.1.a Initial specification of the operational infrastructure (GILABS, M9)

This deliverable will contain the initial specification of the interfaces to be included in the GRAPPLE operational infrastructure. This design document will be the basis for the implementation of the first prototype of the GRAPPLE system (D7.4.a).

D7.1.b Updated specification of the operational infrastructure (GILABS, M18)

According to the cyclical approach proposed for the implementation and integration, the operational infrastructure specification will be incrementally updated during the project lifetime, hence this will be the first update.

D7.1.c Final specification of the operational infrastructure (GILABS, M27)

This deliverable will contain the final and detailed specification of the interfaces to be included in the GRAPPLE operational infrastructure.

2 Introduction

The GRAPPLE system is a component-based service oriented system: the major functionalities are split into modules or components, which in turn are split into services. The purpose of this deliverable is to document the functional architecture of the GRAPPLE system and its integration with existing LMSs in academic and industrial settings. More specifically, it describes the current separation of the whole system into independent components interfaced to external LMSs, and the services that each component offers. Furthermore, it describes the interaction between the components when executing the core functionality of the system.

The operational infrastructure of GRAPPLE includes a set of new GRAPPLE components such as authoring tool set, adaptive engine, user model framework, which are being developed separately. Then this deliverable and tasks need to be kept aligned.

This document starts with a short introduction to the GRAPPLE system, including a short description of the methodology used when building the architecture (Section 3) and a generic system overview (Section 4). The generic system overview (Section 4) also documents the interaction between the components when executing the core system functionality.

Chapter 5 includes detailed descriptions of the identified components. For each of the components, we give a clear description of its role in and contribution to the system, including an outline of its core functionalities. We furthermore describe in detail their constituent services, including, for example, parameters. Finally, for each of the components, we document the expected interactions with the other components and the services they require to function properly.

3 Architectural Approach

The purpose of the functional architecture is to define system components and their interfaces, as well as core data structures for exchange between components.

In a typical software project, development starts with requirements analysis, and only after the first milestone in this activity is achieved, the functional architecture is derived from the captured requirements [10].

GRAPPLE aims at the integration of several newly developed systems and of existing LMSs to one powerful, coherent platform. The whole system is based on several applications that are developed by different parties using different technologies. This leads to the requirement for an extendable and open platform architecture that supports loose coupling of heterogeneous systems.

The operational infrastructure of the GRAPPLE system is developed based on a service-oriented architecture (SOA) [1]. This approach has the advantage that existing, autonomous components can be loosely and flexibly coupled. The usual service implementation nowadays is Web Services, which are also employed here.

Service-oriented architecture (SOA) provides methods for systems development and integration where systems group functionality around business processes and package these as interoperable services. An SOA infrastructure allows different applications to exchange data with one another as they participate in business processes. Service-orientation aims at a loose coupling of services with operating systems, programming languages and other technologies which underlie applications.

This approach allows a relative autonomy of each GRAPPLE component developed in the different work packages of the project. Any work package is responsible for its own contract and public interfaces.

The GRAPPLE project involves several pilot applications such as virtual reality, simulations and possible future extensions that have not been foreseen.

The final version of the GRAPPLE system includes a considerable set of components: the GRAPPLE authoring tool set, adaptive engine, user model framework, etc. are being developed separately. This deliverable and tasks needs to keep them aligned.

4 GRAPPLE Operational Infrastructure Overview

4.1 Scenarios and Use Cases

For presenting the core interaction between the system services a scenario-driven description approach has been selected: the chosen scenarios are taken from the evaluation analysis available in the Requirements Analysis deliverable D10.1 [10].

To initially identify key activities and roles which need to be supported in the GRAPPLE system, three “Learner” scenarios with different levels of complexity and one “Author” scenario have been defined. These scenarios involve learners taking modules in the area of Business English or authors developing modules in this area.

For the first prototype it is not necessary to consider the “Author scenario”: the authoring part is not available at this stage of the project.

Each learner scenario includes a use case diagram, including the identification of the actors and the use cases and an activity diagram showing the components that are available for the first prototype:

- LMS (Learning Management System) in charge of delivering the non-adaptive courses, such as the “traditional” courses provided by the LMS itself, and providing a container for the adaptive courses.
- GALE (GRAPPLE Adaptive Learning Environment) as GRAPPLE Core Engine in charge of delivering the adaptive courses. GALE includes its own authoring component and user model.

4.1.1 Scenario 1 - Learner

- Applied adaptation features: *language*
- Applied adaptation technologies: *adapted presentation* (customisation of the interface)

Lucia is an Italian trainee at the Happy Books company. The adaptive LMS switches the standard language (German) to Italian when she has logged in. In this way, Lucia is able to attend the offered courses in the learning management system in Italian.

Figure 1 shows a simple use case scenario displays the key use cases.

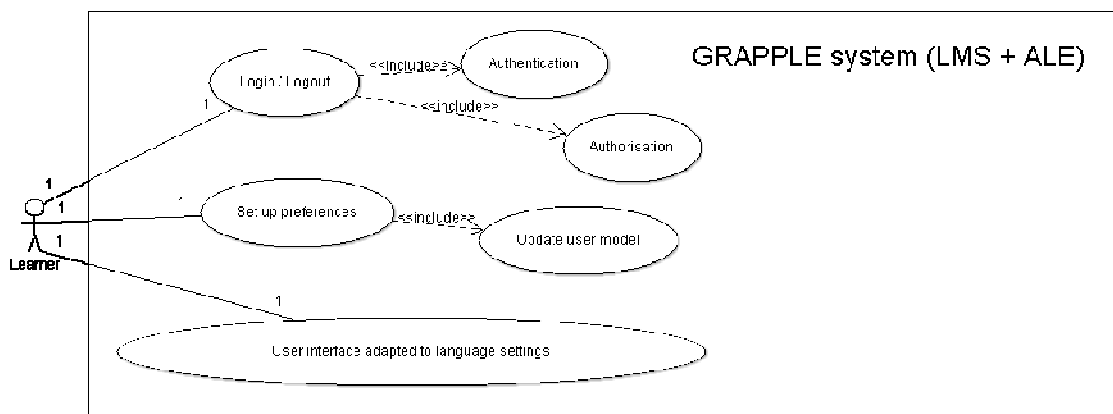


Figure 1: Scenario 1 – Use case diagram.

This scenario has been chosen for its simplicity to illustrate the communication between the LMS and the GRAPPLE system. This scenario can be skipped for LMSs that do not have their own internal User Model.

This use case diagram can be detailed in the following activity diagram:

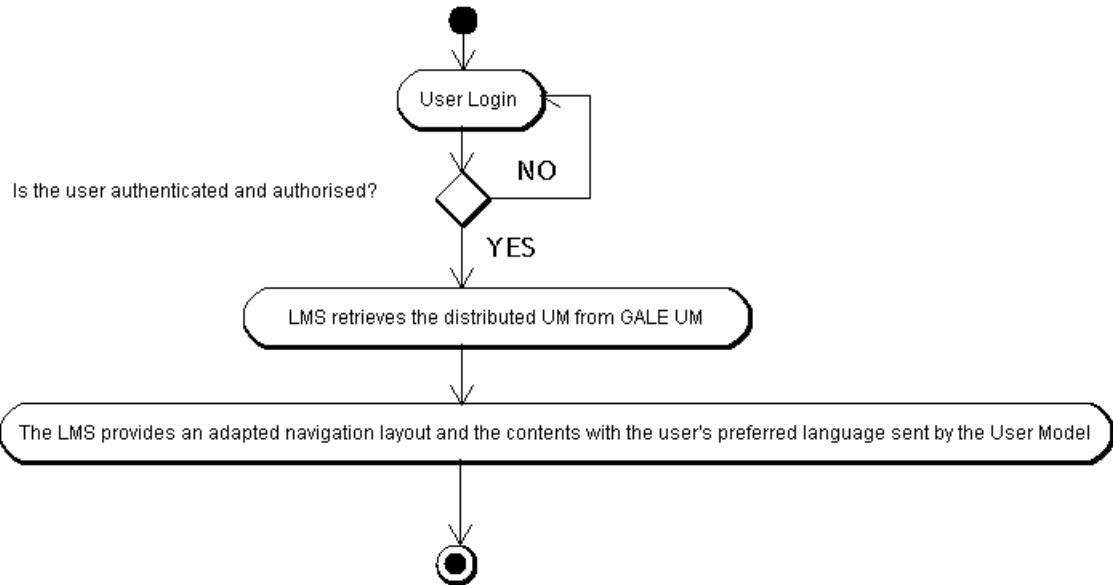


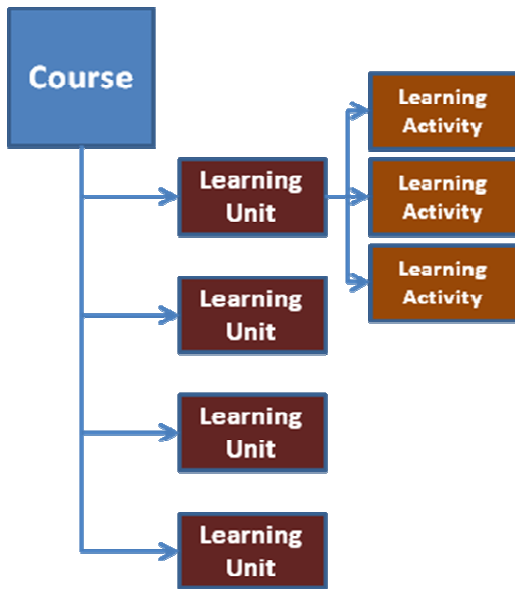
Figure 2: Scenario 1 – Activity diagram.

4.1.2 Scenario 2 - Learner

- Applied adaptation features: *learner goal, learner knowledge*
- Applied adaptation technologies: *adaptive learning activity selection (prerequisites explanations)*

Nicole is taking a course in Business English. Nicole has set communication skills in English as her learning goal. She sets this up in the learning environment. As Nicole has just started the course, the first step she gets presented with a test in order to determine her knowledge level in English. According to the test results, she is at an intermediate level.

The course is composed by different learning units and per each unit we can have several types of learning activity.



In general the learning process is designed in such a way that a learning activity related to the learner goal is presented followed by a test. Nicole clicks on the lesson “Registering” and sees the learning unit “Checking into a hotel”. “Checking into a hotel” belongs to the intermediate concepts, such as “Dealing with customers” and “Send feedback to the office”.

Nicole clicks at the lesson “Registering” and sees the learning unit “Checking into a hotel”. As she wants to improve her communication skills she is provided with a conversation activity. After this a test checks her learning progress. As she passes the test, she continues with the next learning unit “Registering at a conference”. Again she works on a learning activity to improve her conversation skills. The corresponding test doesn’t cause problems for her either.

She continues with the “Dealing with customers” unit of the “Business relations” unit. Again the adaptive system selects the appropriate learning activity and test for her. Nicole’s test results reveal a lack of knowledge with respect to the learning unit “dealing with customers”. Therefore the adaptive system provides prerequisite explanations by presenting both learning units “Negotiating a deal” and “Entertaining at a restaurant”. In this way Nicole can overcome her lack of prerequisite knowledge.

Figure 3 shows a simple use case scenario displays the key use cases.

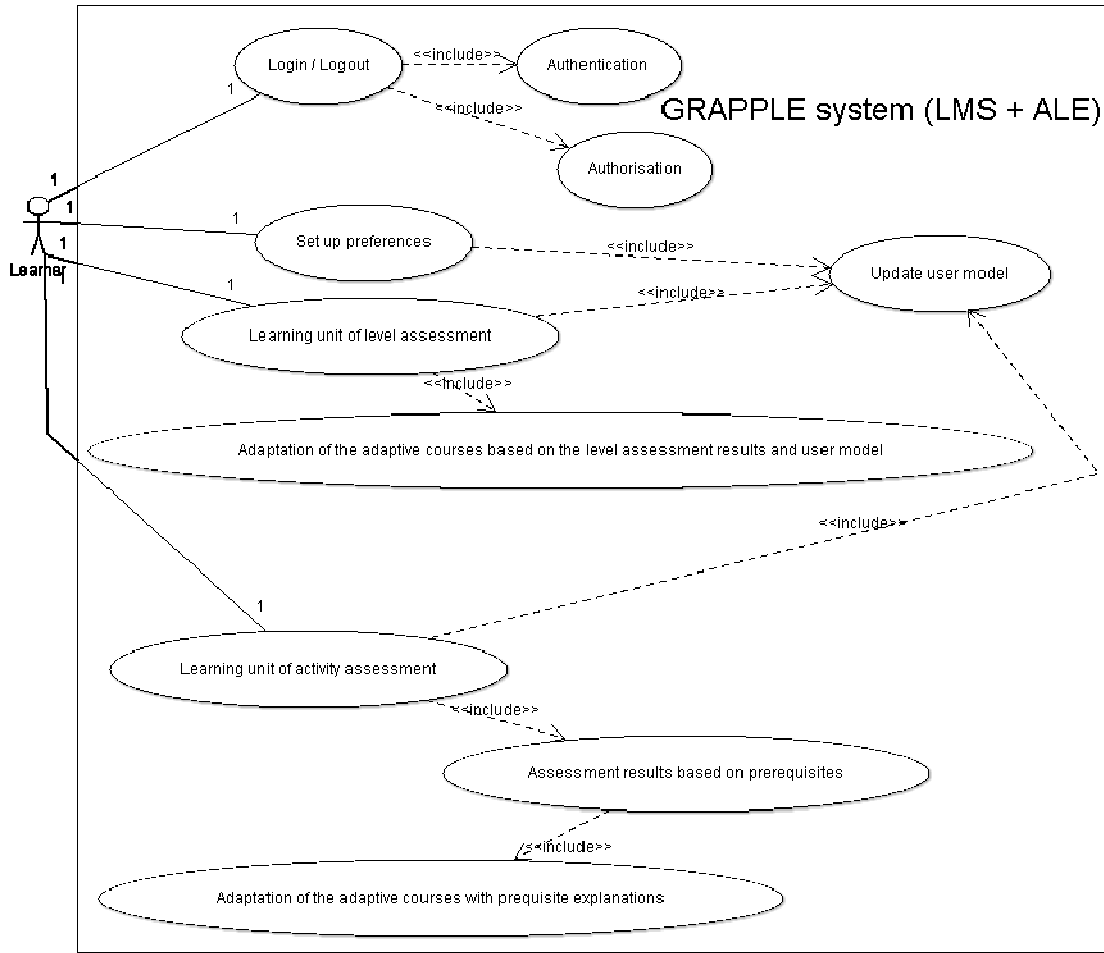


Figure 3: Scenario 2 – Use case diagram.

Figure 4 describes the activities involved in this scenario. In order to make it understandable, the activity diagram has been split into different sub-figures.

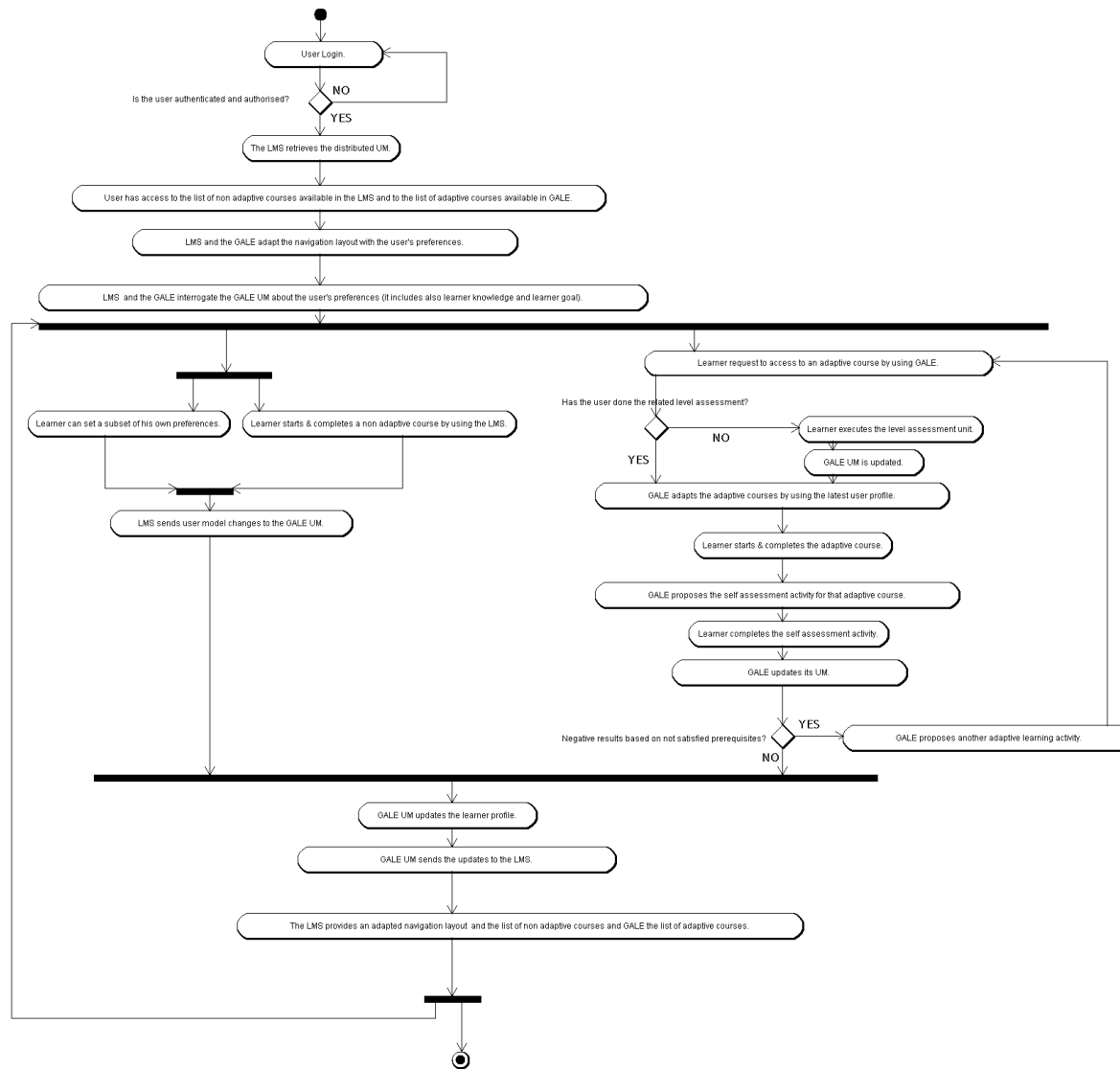


Figure 4: Scenario 2 – Activity diagram.

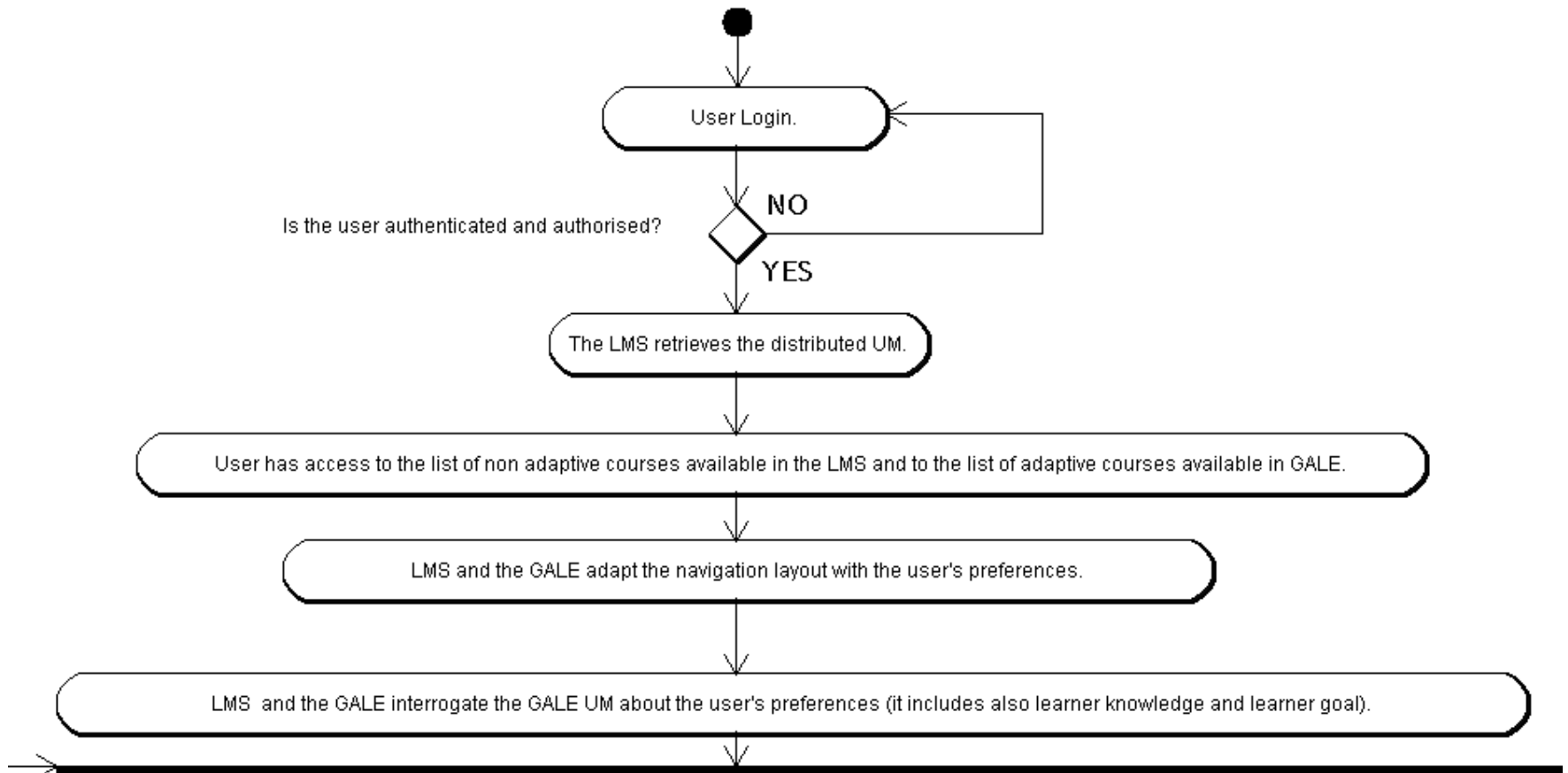


Figure 5: Scenario 2 –Activity diagram – first step.

Figure 5 shows in details the first step of the activity diagram written for this scenario: the user logs into the GRAPPLE system, intended as LMS and Adaptive Learning Environment (ALE).

In this diagram there is the assumption that the LMS has got its own internal UM: **this is not valid for all the LMSs**. In fact not all the LMSs are provided of their own internal User Model. Then, the step “The LMS retrieves the distributed UM.” is optional.

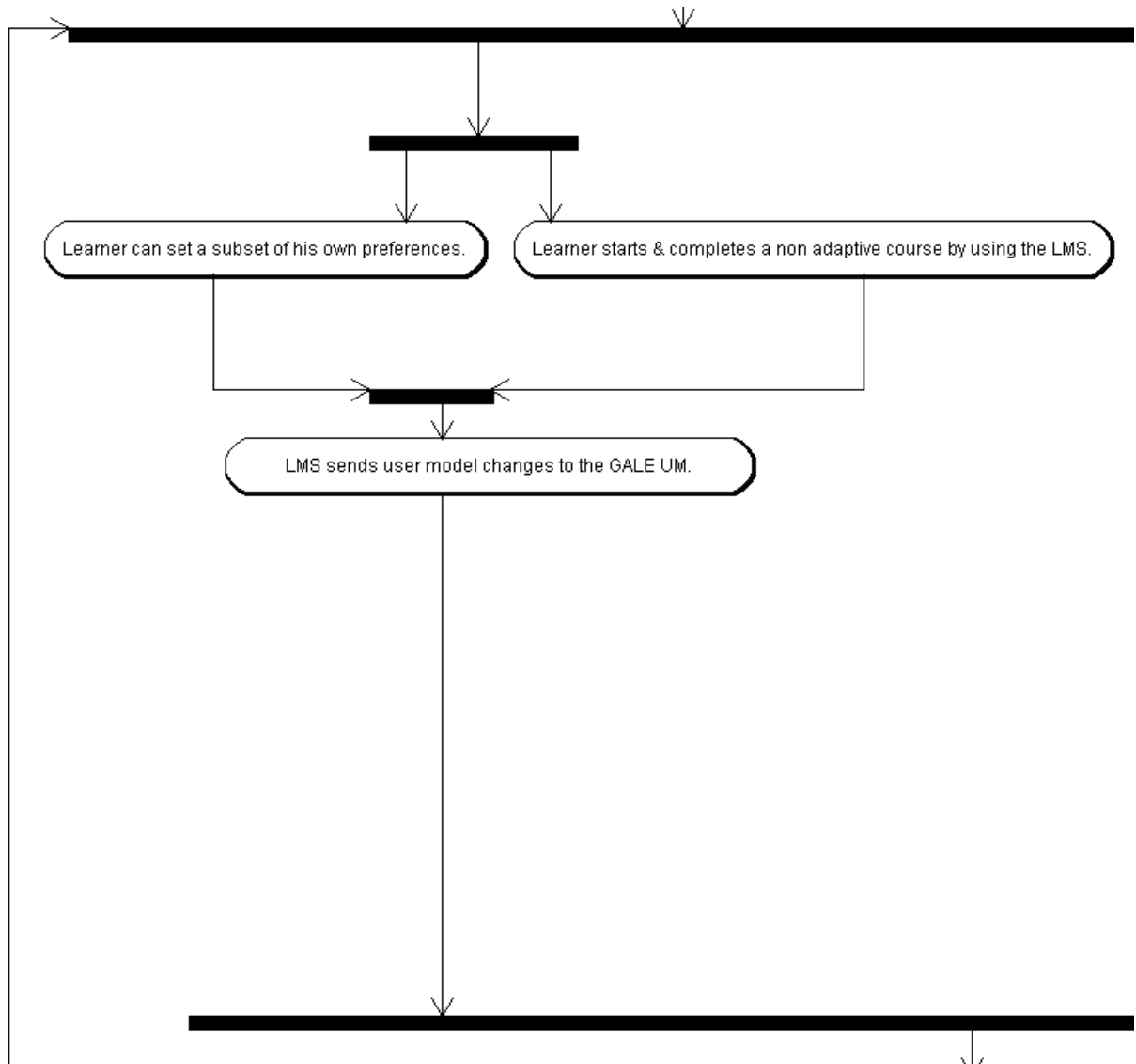


Figure 6: Scenario 2 – Activity diagram – second step.

Figure 6 shows in detail the next steps of the activity diagram for the scenario: the user can execute the following actions:

- The existing LMSs involved in the GRAPPLE project have considerable differences in managing the user details, course information etc. In dependency on the LMS features, the learner can set a subset of preferences
- The learner can start and complete a non adaptive course through the LMS. The scenario requires a level assessment, but not all the LMSs can satisfy this feature. Therefore it is preferable to keep it general and assume the LMS manages internally the learner's level assessment.

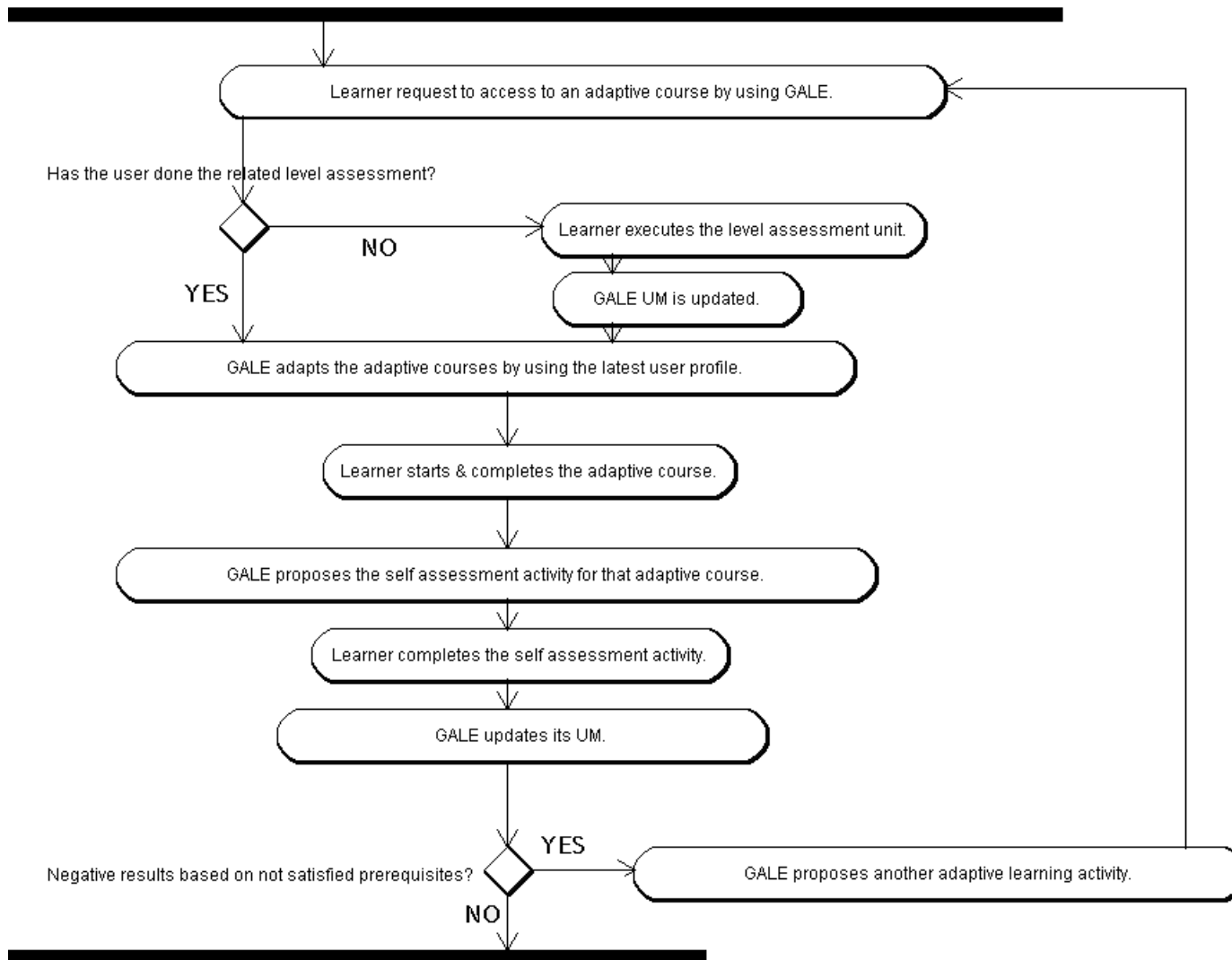


Figure 7: Scenario 2 – Activity diagram – third step.

Figure 7 shows in details the steps of the activity diagram focused on the adaptive courses provided by GALE. First, the learner requests access to the adaptive courses proposed to him based on his user profile. The system deduces the prerequisites for his chosen. In the case where the learner has already undertaken an assessment, GALE can recover this information from its UM; otherwise GALE provides the assessment to the learner. The key part of this diagram is related to the prerequisites evaluation after the self assessment activity: once the learner has completed the adaptive course, the self assessment activity allows the system to understand if the learner needs prerequisites explanations.

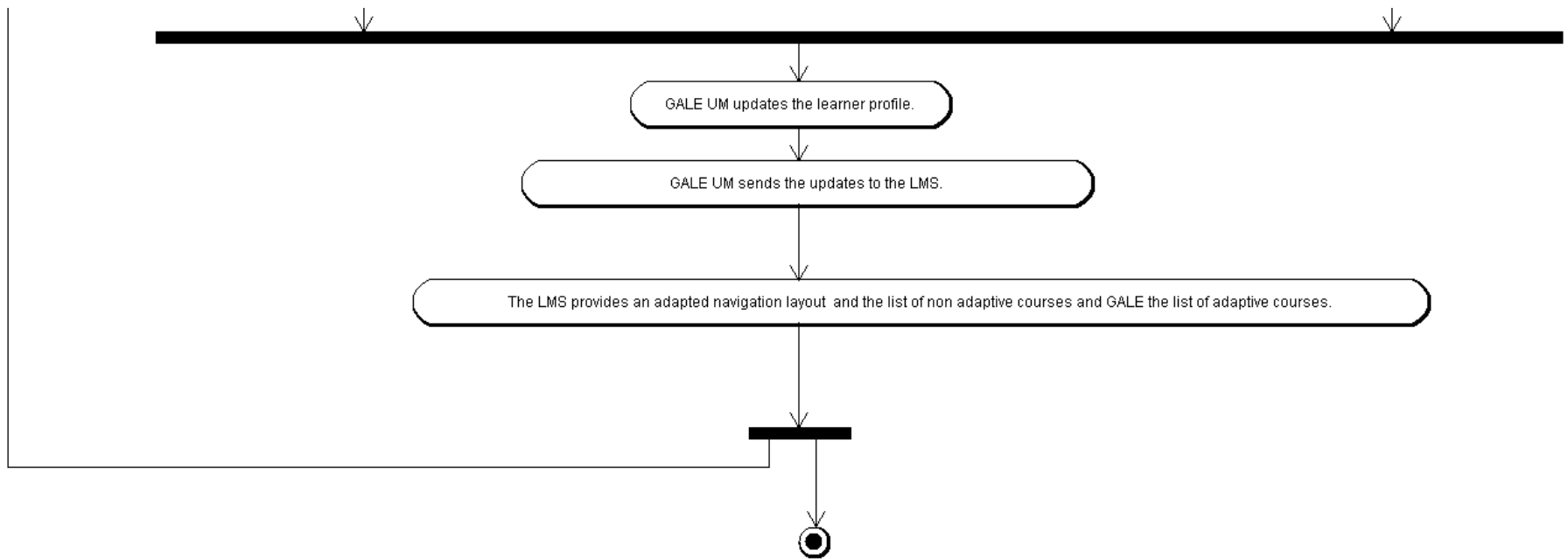


Figure 8: Scenario 2 – Activity diagram – final step.

Figure 8 displays the final step of the activity diagram: in case of adaptive courses or of non adaptive courses the diagram shows that GALE and the LMS need to update properly the learner profile , the navigation layout and the list of the courses available and suggested to the learner. Again, all the LMSs have different features and different architecture: some of them are not able to receive and elaborate any input from its UM. In that case, the update request is ignore.

4.1.3 Scenario 3 - Learner

- Applied adaptation features: *learner goal, learner knowledge*
- Applied adaptation technologies: *adaptive learning activity selection (additional explanations)*

Peter is also attending the same adaptive course. His goal is to improve his listing comprehension. The adaptive system provides the appropriate learning activities. If Peter has not successfully passed the test associated with a learning activity, then as an additional explanation another learning activity is presented to Peter.

This scenario is an alternative of scenario 2: it describes the case of adaptive courses with additional explanations based on the self assessment sections.

Figure 9 shows a simple use case scenario displays the key use cases.

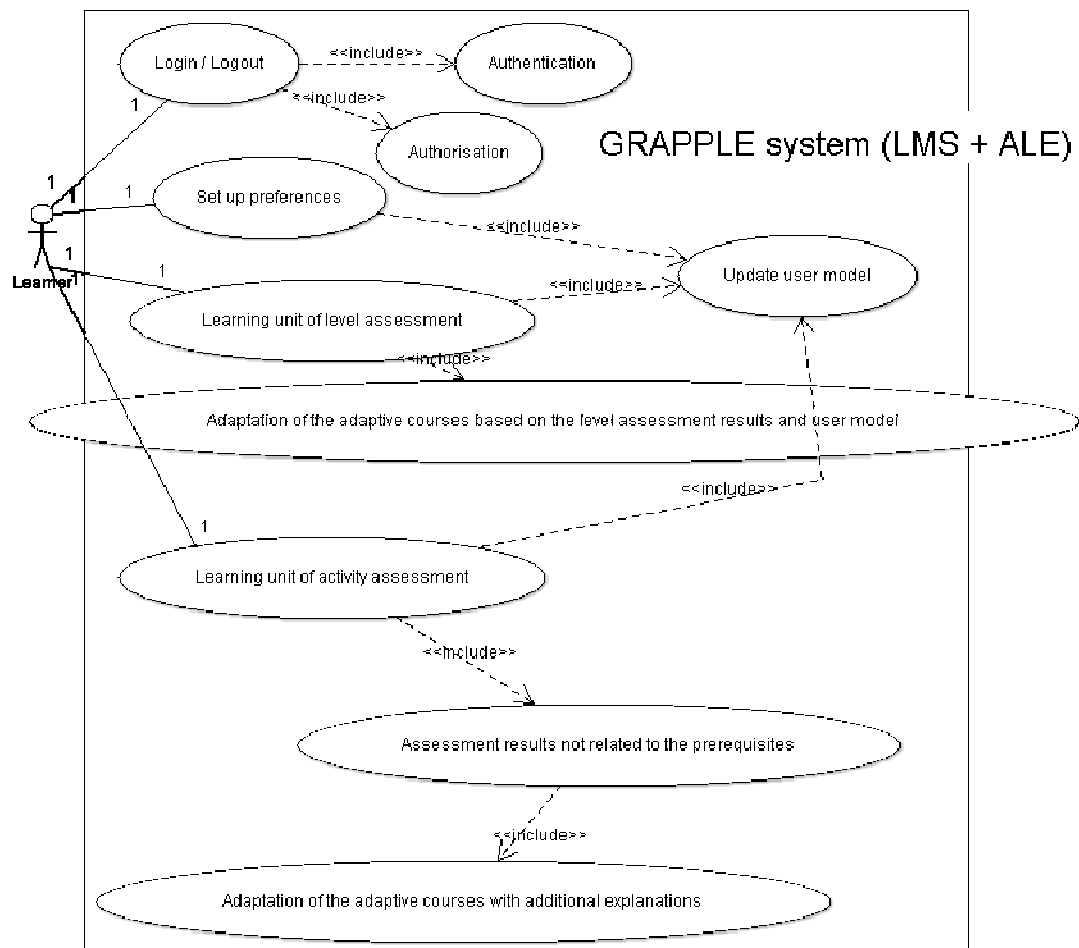


Figure 9: Scenario 3 – Use case diagram.

The activity diagram in Figure 10 and related to this scenario is not very different from the one displayed in Figure 4.

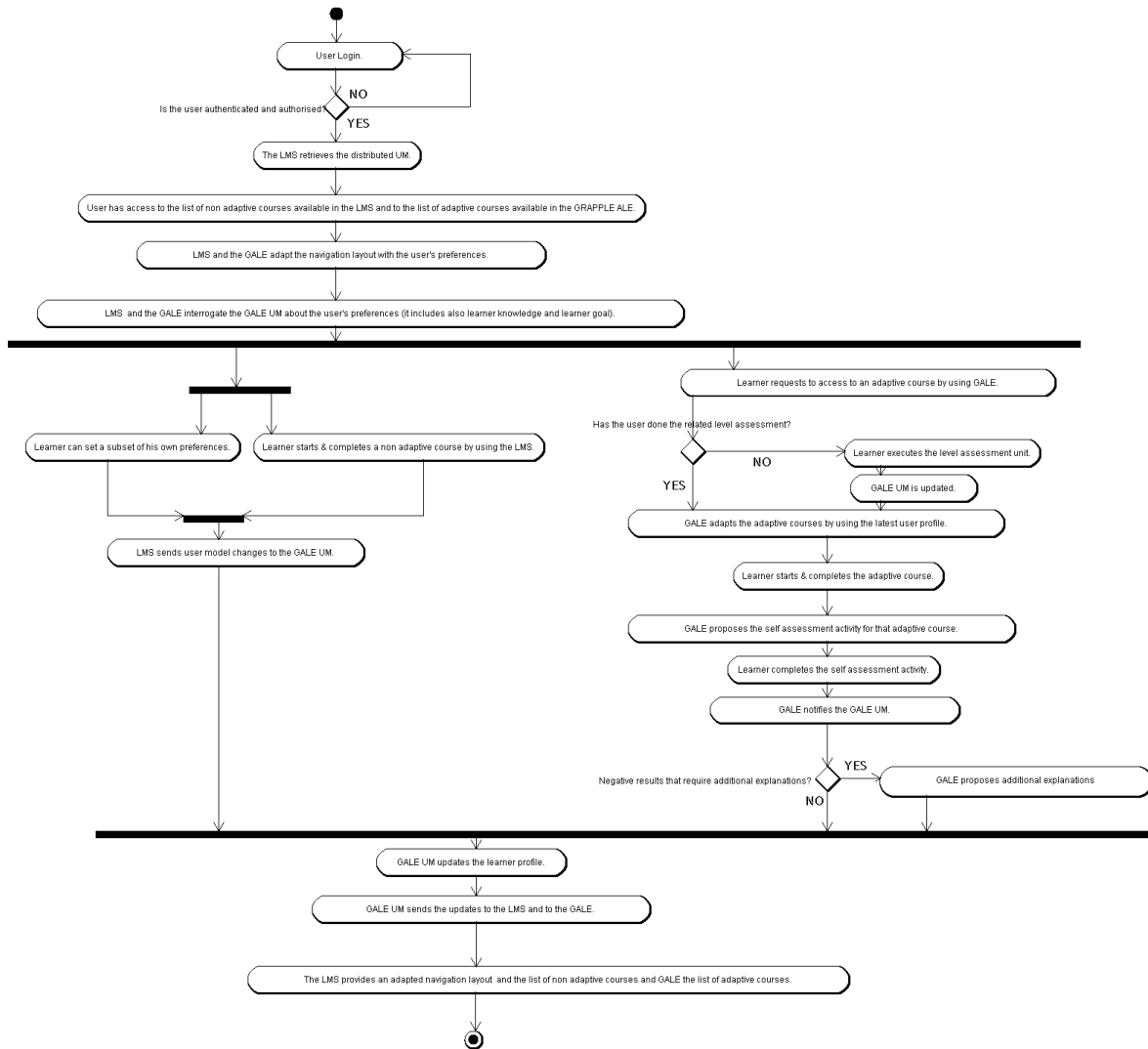


Figure 10: Scenario 3 – Activity diagram.

The only part that differs concerns the adaptive courses, as you can see in Figure 11:

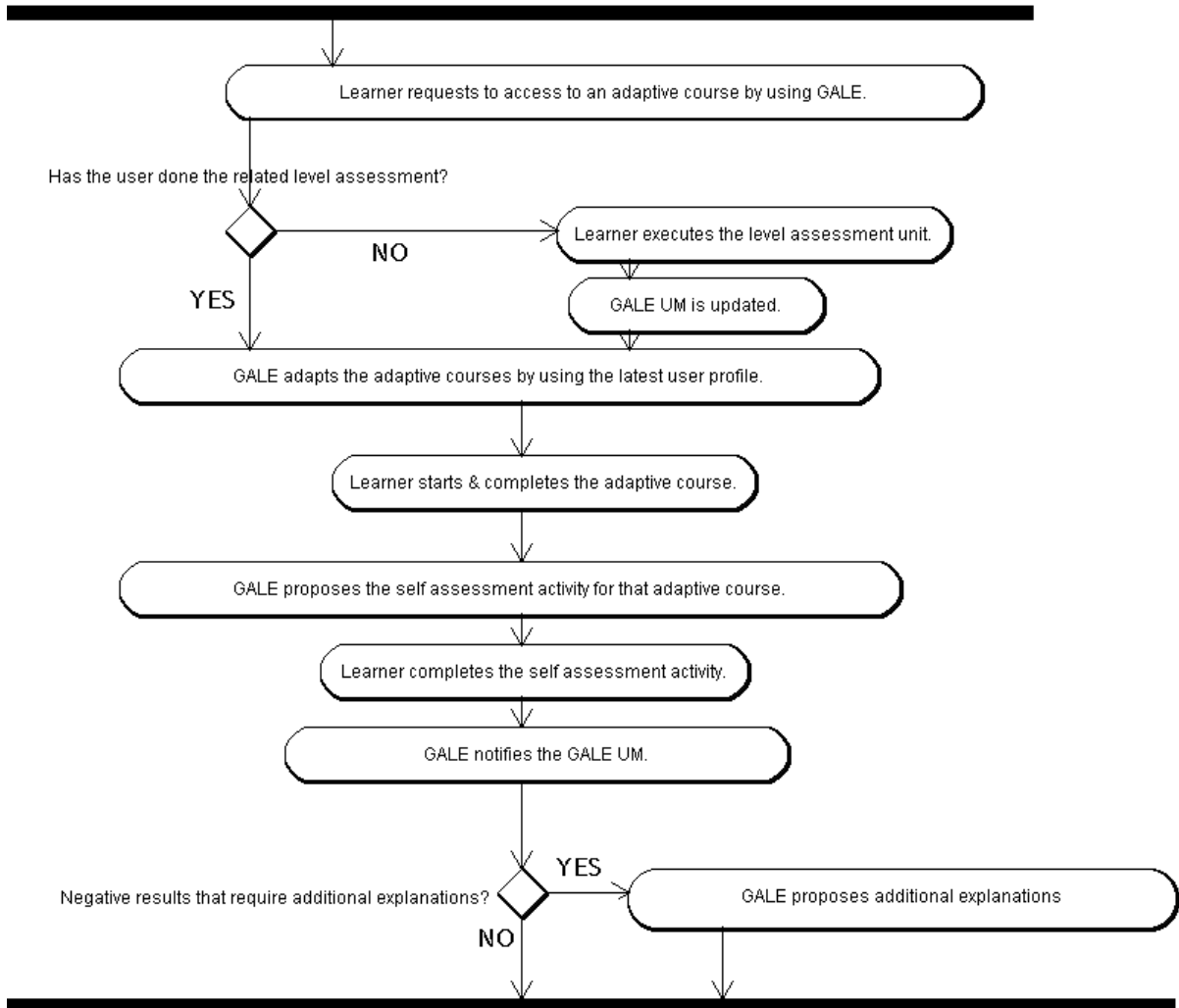


Figure 11: Scenario 3 – Activity diagram – third step.

Figure 11 shows in detail the steps of the activity diagram focused on the adaptive courses provided by GALE: the diagram follows the indications described in the scenario. First, the learner requests to access to the adaptive courses proposed to him based on his user profile, then for that course GRAPPLE system has to evaluate if the learner needs additional explanations. In case where the learner has already executed a level assessment session, GALE can recover this information from its UM; otherwise GALE provides the level assessment session to the learner. The key part of this diagram is related to the evaluation after the self assessment activity: once the learner has completed the adaptive course, the self assessment activity allows GRAPPLE system to understand if the learner needs additional explanations that GALE will promptly provide.

Additional explanations are provided under form of brief description of the points not clear to the learner, and under form of related examples and exercises.

4.1.4 Scenario 4 - Author

Peter has to prepare an adaptive course to teach business English to his students. First he creates the content for the learning units or acquires existing ones. ALE allows Peter to create an adaptive business English course which adapts to a learner's goals, knowledge and language.

Second, Peter sets up the adaptive functionalities with the help of a wizard like interface. He knows that learners with different goals/tasks, knowledge levels, and native languages are attending the business English course. He also knows that these differences include different requirements.

For the creation of the course the following steps have to be performed:

- Author defines a structure or prerequisites between the concepts
- Author defines the connections between concepts (learning activities) and goals/tasks
- Author defines a learning logic (rules for learning process)
- Author defines other rules (e.g. adaptation to user language).

This scenario is present in this document for completeness: the first prototype will not consider this scenario because it does not involve directly the available components for the first integration prototype.

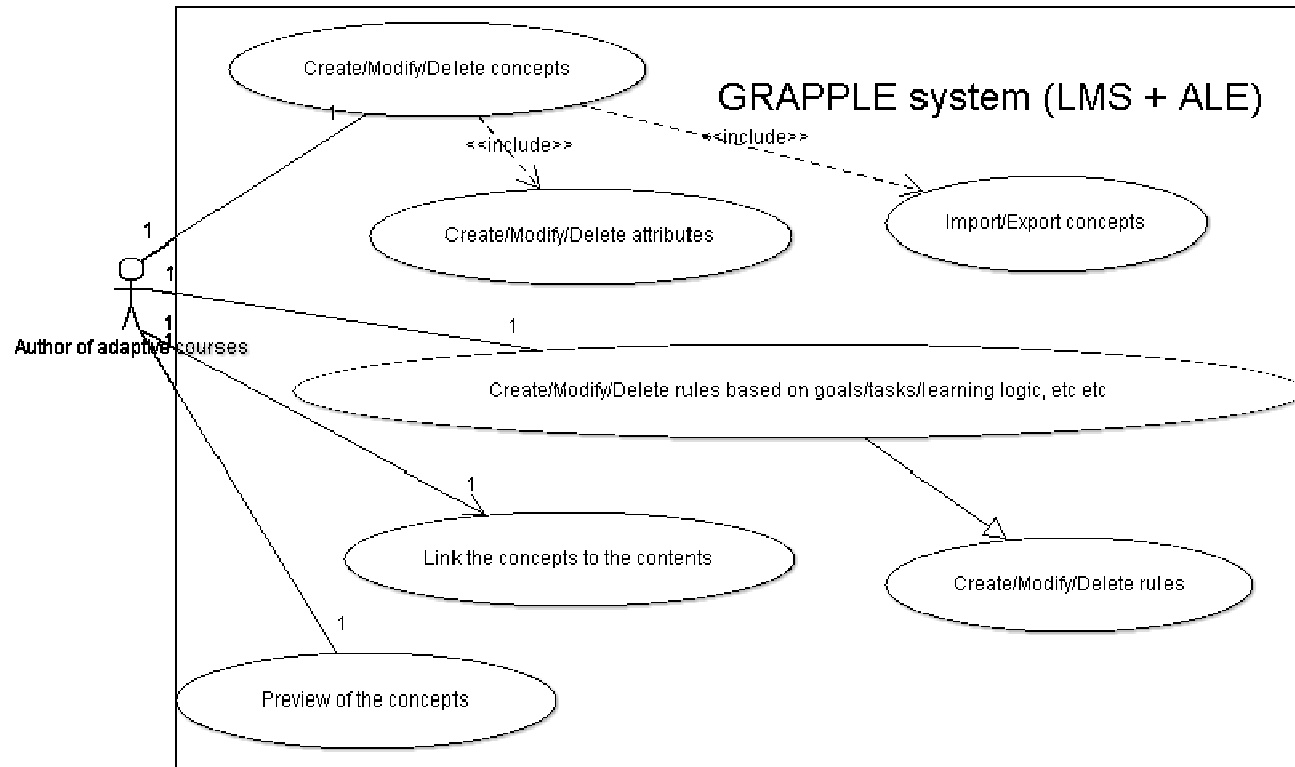


Figure 12: Scenario 4 – Use case diagram.

4.2 Overall Architecture

This section presents an overview of the GRAPPLE system. Firstly the system components are presented and then the core interaction between the components and services are described.

4.2.1 ALE components

This section will provide a short summary of the ALE component which will be integrated into the GRAPPLE system in the first integration prototype:

- Core Engine (WP1): it will access content and user models served from web services or databases, it will perform adaptation by selecting objects, possibly by querying and accessing different sources, and by altering the content of objects (for sorting and link adaptation), and it will talk back to the user model services to inform them about what was selected and presented. This component will be described in [3]. For the first prototype the GRAPPLE Core Engine will be not completed and GALE will be used instead.

4.2.2 LMS

A key factor of the GRAPPLE project is the interoperability between the adaptive facilities developed by the ALE components and LMSs already existing in academic and industrial settings.

The GRAPPLE consortium includes partners who are actively involved with five different LMSs, three are open-source Claroline, Moodle, Sakai and two are commercial learn eXact and IMC CLIX.

The LMSs will provide a container for the adaptive materials provided by the GRAPPLE Core Engine, and in particular GALE. User activities events detection is performed directly by GALE, hence learner profile and UM are updated directly by the Core Engine, not by the LMS.

For the first prototype the adaptive courses are not stored in the LMS databases and are not managed by the LMS itself: LMS provides the user profile information and the access to the non adaptive courses. Then GALE is running in a frame or new window of the LMS.

Some LMS may not have an integrated local UM. In this case, some (or even all) events propagated by the GALE UM to the LMS may be discarded by a particular LMS because it does not have a local interpretation and storage for particular UM data.

The five LMSs involved in the GRAPPLE consortium are very different:

- Claroline and Moodle are PHP based
- Sakai and IMC CLIX are Java based
- learn eXact is .NET based.

The different technologies employed by these LMSs require the formalisation of the WSDL to evaluate the impact on all the LMSs.

Furthermore, all the LMSs partners indicated the possible risk of slow performances: there could be a limited responsiveness of the system due to unavailable Web services or slow response of a Web service employed in the Event Bus in case of synchronous communication.

In Appendix there is a short description of the LMSs involved in the project.

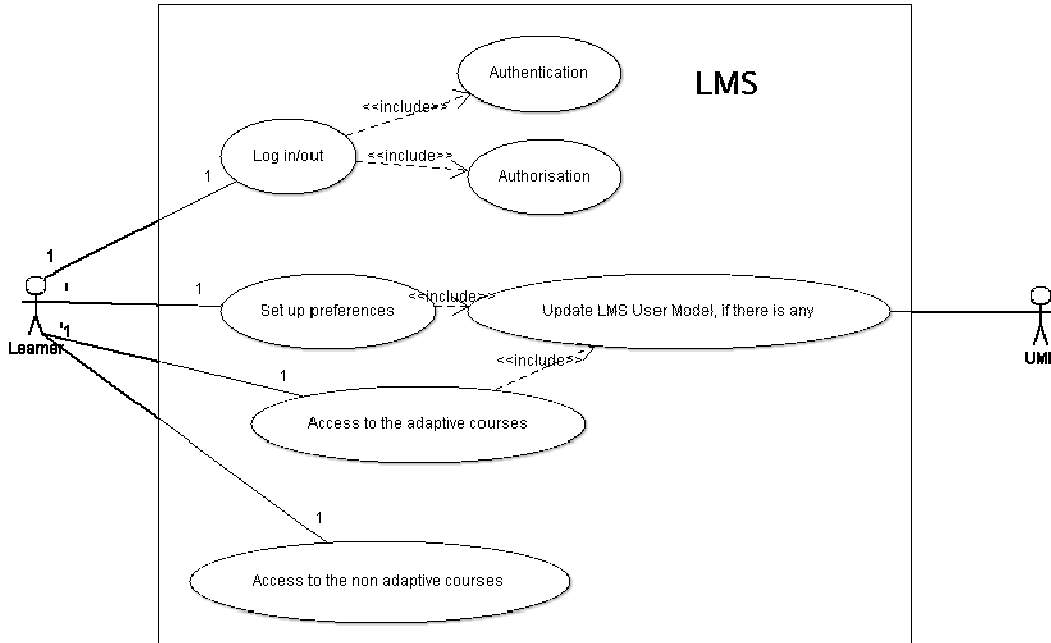


Figure 13: LMS – Use case diagram.

4.2.2.1 Services offered to other components

Each LMS implements an interface to allow external entities like the GALE UM to update the UMs.

The interface could provide an operation like:

- **updateUserModel(user,...)**: it will allow the LMS to update different parameters stored in the GALE UM.

4.2.2.2 Services required from other components

The services required by the LMS are two:

- User model notification channel: the LMS needs a channel to notify the changes by the learner to his own User Model. The GALE Event Bus described in [3], will provide that.
- Adaptive Learning Material Presentation: the LMS needs an external tool to deliver the adaptive learning material. To perform that, GALE gets a frame within the LMS.

4.2.2.3 Interaction between the ALE components and LMS

By using the above described scenarios it was possible to identify the type of interaction among the components.

Figure 14 gives an overview of GRAPPLE components needed for the first integration prototype to satisfy the three learner scenarios:

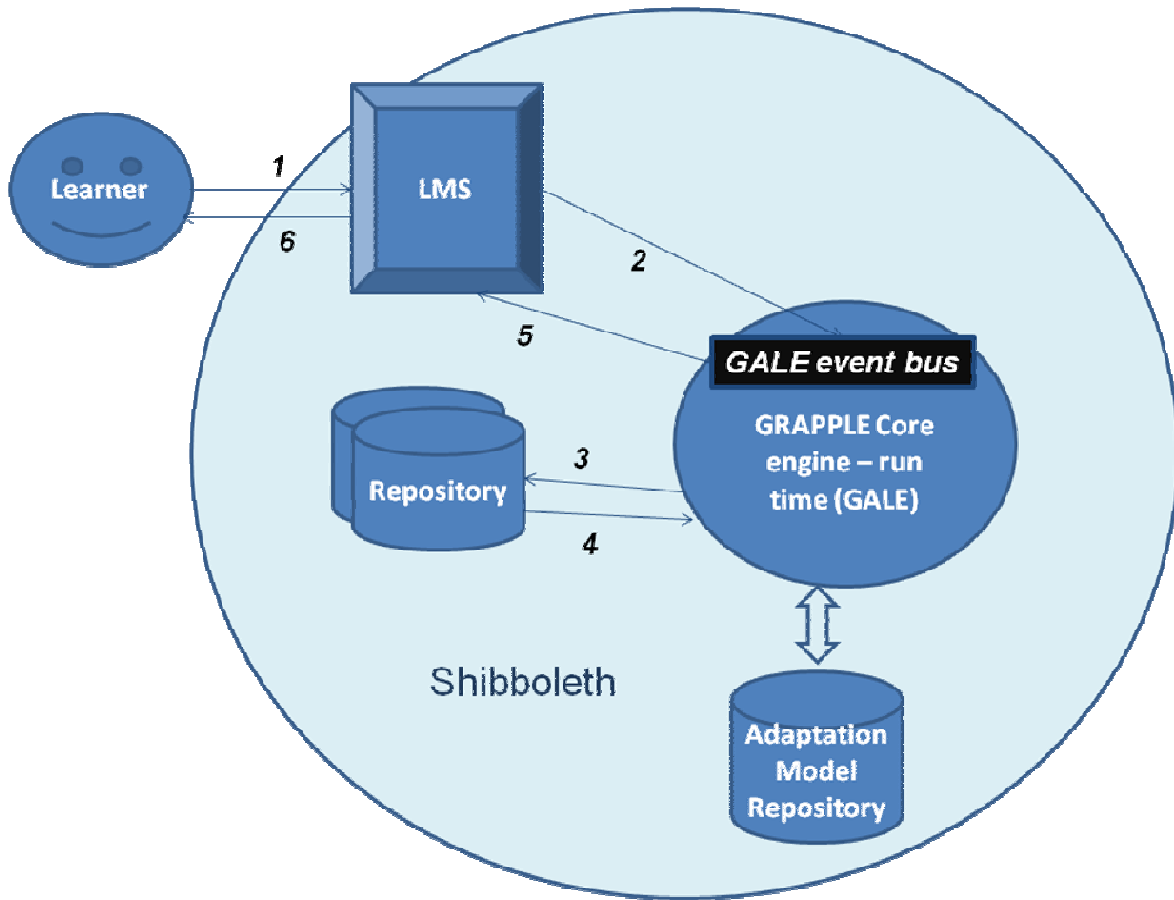


Figure 14: Learner scenarios – first prototype.

1. The learner accesses the LMS: the SSO facility allows the learner to sign in simultaneously to the LMS and to the GRAPPLE system.
2. The LMS communicates to the GALE Event Bus to update the GALE UM by using the data of the learner has just logged in, and triggers the GALE.
3. The GALE retrieves the adaptation models stored in its dedicated database, recovers all the necessary user data from its own GALE UM and interrogates the courses repository(ies) to retrieve the requested and/or appropriate course.
4. The courses repository(ies) gives access to the requested and/or appropriate courses.
5. The GALE adapts the courses and their related presentation by using the adaptation models and the user data and sends them to the LMS.
6. The LMS presents and provides the course adapted to the characteristics specific of the learner.

Error! Reference source not found. gives an overview of the GRAPPLE components needed for the author scenario. The most of the components necessary for the author scenario are not available for the first prototype and then it will be concretized essentially by the GALE component as shown below:

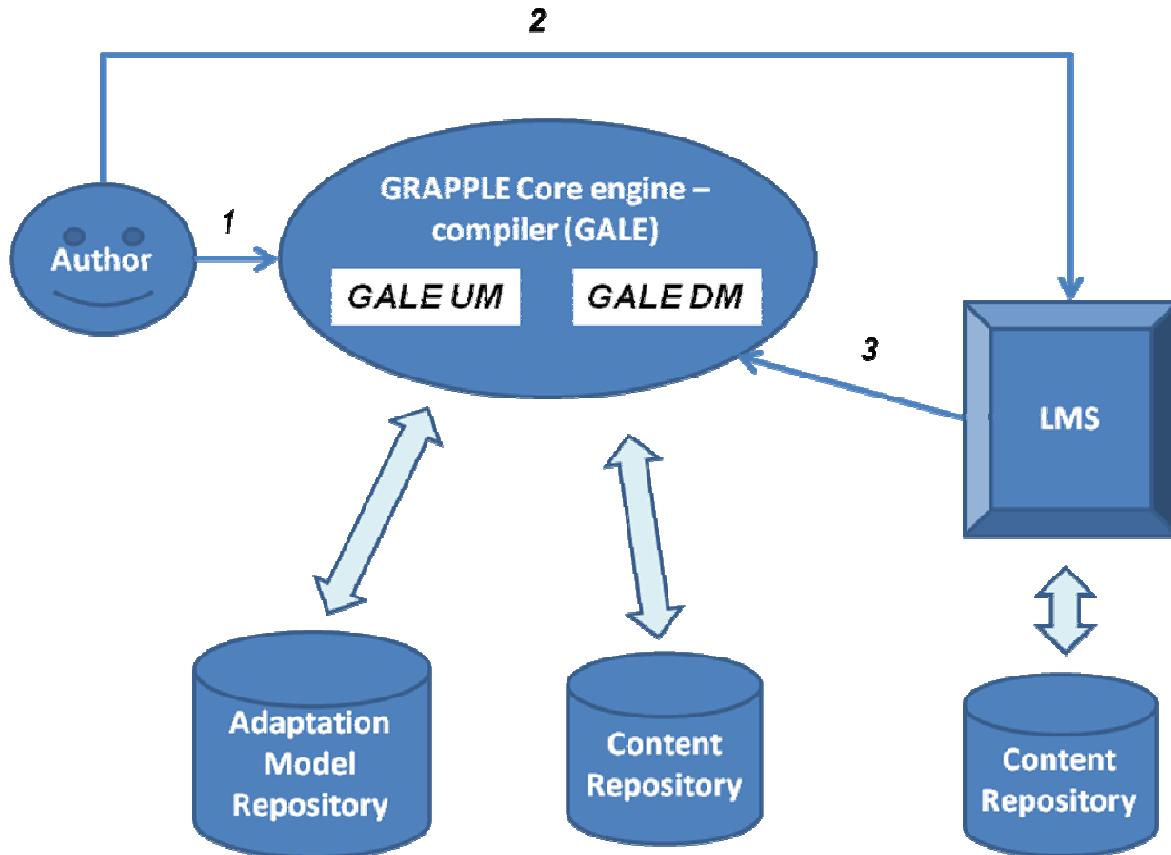


Figure 15: Author scenario – first prototype.

1. The author accesses to the GALE to define and manage the concepts necessary to his adaptive courses, to create and manage the rules/relationships that rule his adaptive courses, and to build the adaptive courses that will be available in a Content Repository.
2. In order to make the adaptive courses available to the Learners by using the LMS, the author accesses to the LMS
3. And he has to make the single adaptive courses available from the LMS (by using the URL).

4.2.2.4 Single sign on facility

SSO is a mechanism whereby a single action of user authentication and authorization can permit a user to access all computers and systems where he has access permission, without the need to enter multiple passwords. SSO makes the system more usable for the authors and learners and provides easier management of access privileges to various resources. There are many SSO mechanisms and not all of them are supported by all the LMSs present in the consortium.

First the choice was to use Shibboleth justified by using a standard instead to create a new functionality. Shibboleth is supported by three of the five LMSs involved in the GRAPPLE project: Claroline, Moodle, Sakai and IMC CLIX.

This choice is still under discussion.

The Shibboleth® System is a standards based, open source software package for web SSO across or within organizational boundaries. It allows sites to make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner.

The Shibboleth Architecture extends the SAML 1.1 SSO and attribute exchange mechanisms by specifying service-provider-first SSO profiles and enhanced features for user privacy. Scavo [2] provides a technical overview.

The functional components of a conforming Shibboleth implementation include an identity provider, a service provider, an optional “Where are you from?” (WAYF) service, and various interacting subcomponents. For the GRAPPLE project WAYF service appears unnecessary.

There are three primary parts to the Shibboleth system:

- Identity Provider - the software run by an organization with users wishing to access a restricted service;
- Service Provider - the software run by the provider managing the restricted service and secure resources.
- One or more LDAP repositories – they could be distributed among different institutions and are the source for the personal accounts.

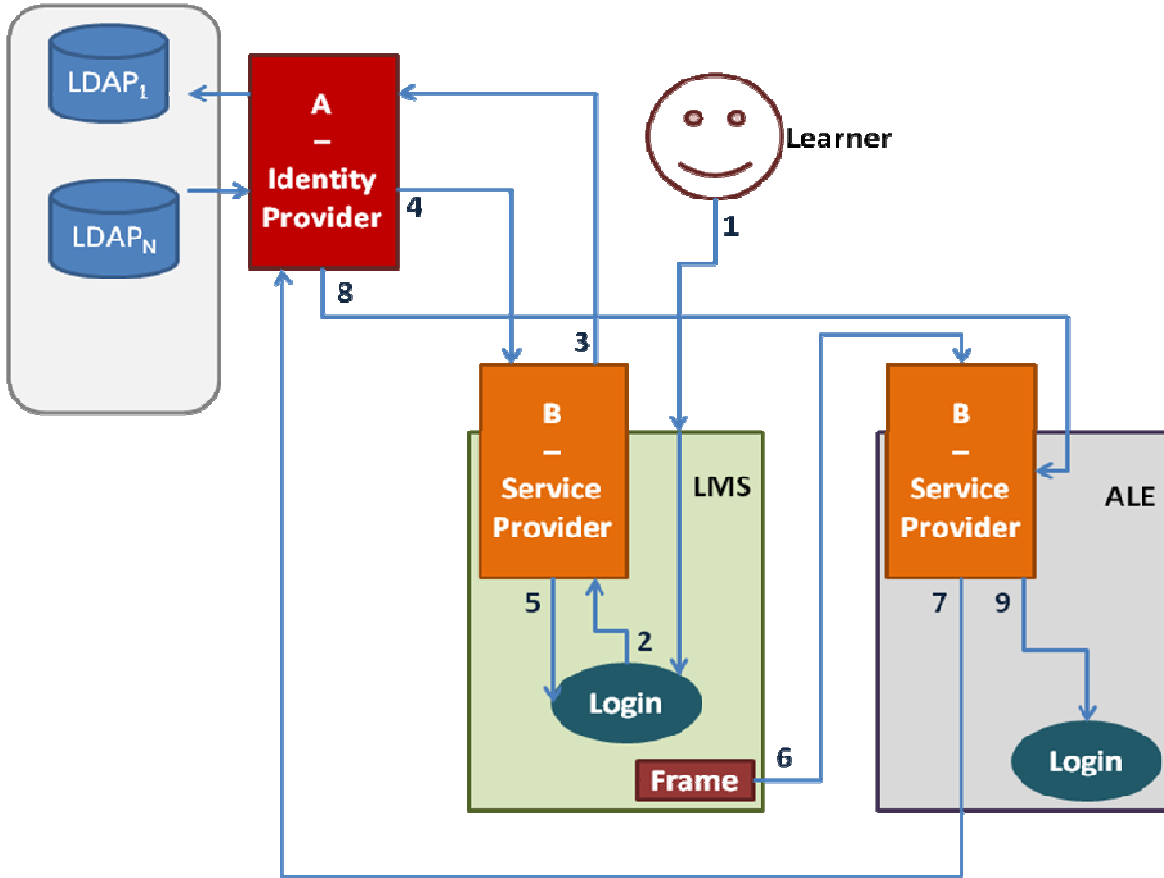


Figure 16: Shibboleth & GRAPPLE.

where A is **Shibboleth Identity Provider** and B is the **Shibboleth Service Provider**.

Pre-condition: User not already logged in with Shibboleth.

1. The Learner requests to access to the LMS
2. LMS asks the local Service Provider
3. The LMS Service Provider asks the global Identity Provider with the provided credentials
4. The global Identity Provider replies to the LMS Service the Learner has requested to access
5. LMS guarantees the Learner the access to the resources
6. Request of access to ALE for the adaptive contents inside the LMS, through a Frame
7. The ALE Service Provider asks access to the global Identity Provider by using the token already provided



8. The global Identity Provider remembers the Learner has already logged in
9. The ALE Service Provider guarantees access to ALE to the Learner.

5 GRAPPLE key components – first prototype

Figure 17 displays the GRAPPLE components and the communication among them. It is possible to identify a key element, the GALE Event Bus described in [3] that handles the connections among the LMS and GALE, the GRAPPLE Core Engine of the first prototype.

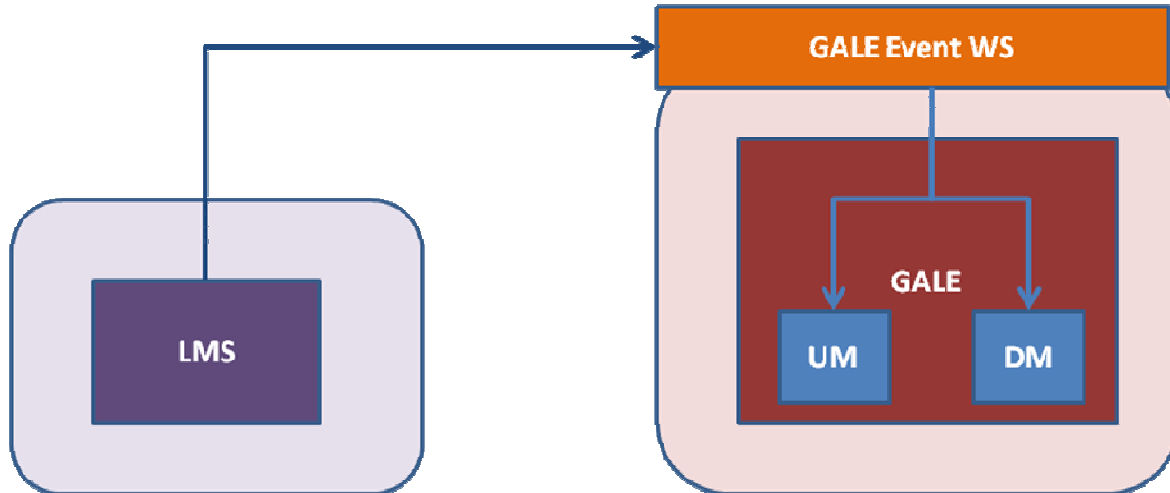


Figure 17: GRAPPLE first prototype architecture.

Components send requests to the GALE Event Bus, and other components may listen to these events and handle them. This may result in a reply sent to/through the GALE Event Bus as well.

In Figure 17 it is possible to see the communication among the main GRAPPLE components.

Each arrow represents a service provided by the target component to the source component of the arrow: the GALE component makes available the GALE Event Bus collects all the events sent by the LMS and generates events that are handled internally in the GALE component.

5.1 GRAPPLE Core Engine - GALE (WP1)

5.1.1 Functionality

One of objectives of WP1 is to develop a Core Engine that can access content objects and user models served through web services or databases, that works as a web service itself, and that performs adaptation by selecting objects, possibly by querying and accessing different sources, and by altering the content of objects (for sorting and link adaptation).

The Core Engine needs to be able to generate a “final” presentation (including a HTML frame layout, and navigation aids), but also needs to be able to simply parse and adapt content as part of a delivery pipeline in which it is not the final step.

For the first prototype GALE will be used as GRAPPLE Core Engine.

5.1.1.1 GALE Event Bus

Communication within the GRAPPLE framework is facilitated by the Event Bus. The Event Bus supports a number of services that should be used by all communicating applications that use the GRAPPLE Framework, i.e. the GRAPPLE Core Engine, and the LMSs.

The event bus is a web service that exposes its functionality through SOAP. Events on the event bus consist of a method-name (string), and a list of parameters (also strings). The event bus maintains a list of EventListeners. An EventListener has a single method called 'event' that takes a method-name and a list of parameters. The result is a list of strings. When EventListener's are registered on the event bus, the event bus sends a register event to the new EventListener with its own address as a parameter. The register event should return a list of methods that the EventListener wants to listen to. When the event bus receives an event, it will call all registered EventListener's, merge the answers and return that as its answer.

5.1.1.1.1 The GALE DM Service

The adaptive engine and the UM service of GALE need domain model information. They request information by sending a 'getdm' event to the event bus. This method should be supported by at least one service on the event bus. The default DM service of GALE listens to this event and returns the requested information. It uses a database and Hibernate (www.hibernate.org) to store and retrieve information. The 'setdm' method is also supported, to allow changes and additions to the domain model.

Beside the default domain model service, GALE includes an AHA! 3 domain model service. This service only supports the 'getdm' method and uses .aha files stored in the \$AHA_HOME/config directory as its source of domain model information. The adaptation engine does not know where the data comes from when requesting domain model information. This is all handled by the event bus.

Domain model services are expected to post updates on the event bus when their content changes. They should use the 'updatedm' event to indicate such a change in content.

Below is a more detailed description of the events generally supported by domain model services (an 'l' behind the name indicates the service listens to this type of event, an 's' behind the name indicated the service sends this type of event):

Method	Parameters	Result	Description
getdm (l)	application:<app-name> or concept:<concept-name>	A list of the serialized entities that are part of the specified application or concept.	Retrieves domain model information.
setdm (l)	A list of the serialized entities that should be set. May also include the special 'remove-app:<app-name>' EventHash.	'result:ok' if the operation succeeded. 'result:<exception-class>' if the operation failed.	Sets domain model information.
updatedm (s)	A list of the serialized entities that changed in the domain model.	Ignored.	Notifies listeners on the event bus that the domain model has changed.

The default domain model service sends 'updatedm' events whenever changes made by a call to 'setdm' have resulted in a change to the database. The AHA! 3 domain model service sends an 'updatedm' event for a particular application whenever its underlying .aha file changes (monitors the file system).

5.1.2 The GALE UM Service

The adaptive engine part of GALE depends on UM services on the event bus to provide user information. The user model events are split into 'um' events and 'entity' events. The entity events communicate information about the actual user or group and its properties. Properties can be a password, whether the user is a regular user, author, administrator, or a combination of these, etc.. The UserEntity maintains a list of 'child' entities and one possible 'parent' entity (the group). This information can all be set and retrieved via the 'setentity' and 'getentity' events.

The actual variables that comprise the user model data are set and retrieved via the 'setum' and 'getum' events.

Below we give a more detailed description of the events generally supported by user model services (an 'l' behind the name indicates the service listens to this type of event, an 's' behind the name indicated the service sends this type of event):

Method	Parameters	Result	Description
getentity (l)	entity:<entity-name>	The serialized UserEntity found in the database.	Retrieves UserEntity objects from the database.
setentity (l)	The serialized UserEntity that needs to be changed.	'result:ok' if the operation succeeded. 'result:<exception-class>' if the operation failed.	Sets UserEntity objects in the database.
getum (l)	application:<user-name>.<app-name>	All serialized user model variables that are part of the specified application and their values.	Retrieves user model variables from a specific application.
setum (l)	All serialized user model variables and values that should be changed.	'result:ok' if the operation succeeded. 'result:<exception-class>' if the operation failed. Followed by a list of all user model variables whose value changed because of the updates (including requested changes).	Updates user model variables.
updateum (s)	A list of user model variables and their values that have changed.	Ignored.	Notifies listeners on the event bus that the user model has changed.
updatedm (l)	A list of the serialized entities that changed in the domain model.	'result:ok' if the operation succeeded. 'result:<exception-class>' if the operation failed.	Updates the internal cache and recalculates non-persistent attributes whose default code has changed. May result in updateum events.
queryum (l)	query:<hibernate-query-string>	A list of serialized user model variables and/or UserEntity's that result from executing the specified Hibernate query.	The 'nl.tue.aha.common.data' package specifies a set of Hibernate enabled classes that are used to persist data to a database. Any query based on these classes can be used to retrieve data from the UM service.

The adaptation engine part of GALE will request a specific user model part (the part identified by the unique username and an application name) only once. The values retrieved are cached and only updated directly in the cache or by listening to 'updateum' events on the event bus. Hence there is no need to request the information more than once.

6 Appendix

6.1 CLAROLINE

6.1.1 Description of the LMS

Claroline is an Open Source elearning platform widely used. It is translated in more than 35 languages and use in more than 1250 universities, high schools, and enterprises located in more than 100 countries.

Claroline's best advantage is its simplicity. It is as been made as intuitive as possible to avoid requiring teachers and learners to need a formation before they can start using the LMS. In other words teachers should be able to put a course online (on intranet or internet) fast and without having to read manuals or documentation.

Claroline Learning Management System is built on courses that teacher can create. Each course is defined by a code and a name and can use a different language. Several tools are provided to the course manager to let him put its pedagogical content online.

Default Claroline tools in each course:

- Course description
- Agenda
- Announcements
- Documents & links
- Exercises
- Learning path
- Assignments
- Forums
- Chat
- Wiki.

At this time Claroline supports the following IMS standards : SCORM 1.2, SCORM 2004 (partial support) and IMS/QT1 (for a subset of question types).

Beside pure eLearning usages, Claroline is also often used as a collaborative space for project management or communication.

External authentication is available through a flexible driver-based architecture that allows users to connect to Claroline if they are known on another application such as LDAP, another LMS (Moodle for example) or another Web Application like SPIP.

SSO is also available using Shibboleth or CAS.

6.1.2 Integration with GRAPPLE ALE

As only high level events should be implemented Claroline will send events like exercise done or learning path module completed to the GRAPPLE EventBus using a PHP SOAP Client library like php_soap or NuSOAP.

Any other high level event can be added on demand. Lower level events could also be sent to the Event Bus if required.

In addition, Claroline could made informations about users available through a SOAP server.

6.1.3 Limitations and Risks

As for any web service based application Claroline will hang on until the SOAP request is terminated. So it may slow down the application if for any reason the Event Bus cannot answer fast.

6.2 MOODLE

6.2.1 Description of the LMS

Moodle is an open source LMS/ Course Management System based on a constructivist and social constructionist approach to education. The name stands for Modular Object-Oriented Dynamic Learning Environment.

Moodle was first released in the 1990s, when a WebCT administrator (Martin Dougiamas) started to develop some tools to complete the platform: after some time he decided to develop a new LMS and released the first public version at the end of 1990.

After that, Moodle has been evolving since 1999 (since 2001 with the current architecture) and in the version 1.5 (May 2005) some major improvements in accessibility and display flexibility were developed. The current version is 1.9, which was released in March 2008. Version 2.0 is in beta phase and should be released in short time. Due to its diffusion, a large community of users and developers has grown around Moodle's website (<http://moodle.org>); Moodle has been translated into more than 70 different languages. More than 300 add-ins, hacks, plug-ins and 50 themes are freely available on the website.

Some number on moodle diffusion, based only on institution that decided spontaneously to publish their data: more than 50.000 registered installation with almost 2,5 millions of course and more than 25 millions of users, supported by 2 millions of teacher. (data from <http://moodle.org/stats/>)

The support for standards is: SCORM 1.2, IMS LD (almost Level A), IMS QTI, (probably IMS CP, but to discover real compatibility).

The modularity of the platform allows a quite easy way of extend it. In Moodle, several plug-in API allows an easy and maintainable way to add new functionality to Moodle. Two main types of extension are available: blocks and modules.

The block is useful to implement a plug-in outside the main lesson space, while the module is integrated in the course content, and used by students, as part of teaching materials. For instance, we could integrate in moodle a new block to send SMS to other students, or we could add a new module that provides students with a new type of quiz activity.

It is possible to develop plug-ins in order to integrate external programs or services into the learning management system: the level of integration could vary from the most simple, such as only a common web interface to more complex approaches. In recent releases it is possible to develop a web services interface for the Moodle functions, in order to offer a more standard way to integrate it with external programs.

6.2.2 Integration with GRAPPLE ALE

Moodle (PHP) already collects user interactions data in the form of logs. Technically, it is relatively easy to send log data to an external repository or application, in the form of "*the user X at the date D and time T has loaded the resource Z*" or "*the user X at the date D and time T finished the quiz Z with the final grade W*". In the current version of Moodle, it is not possible for an external entity to subscribe to a Moodle service and request the tracking data automatically. However, such a functionality can be implemented, if necessary.

There's no mechanism inside Moodle to extract new information from log data. For instance, at the moment, it is not possible to derive new information such as "*the user X has achieved the Y level of mastery on the subject Z*".

6.2.3 Limitations and Risks

The internal 'user model' in Moodle consists of personal and contact data only. Nevertheless, in the next version (2.0) a quite sophisticated outcome matrix for learning activities is provided, hence should not be problematic the integration with the GRAPPLE UM. Probably there could be the necessity that the process of collecting data for modeling high level actions able to influence user profile (such as time spent on a resource, or learning path followed) should be tracked by an ad-hoc engine and sent to the GRAPPLE architecture, due to the fact Moodle itself doesn't collect this type of information.

All the usual problems correlated to web-service asynchronous invocations should be taken into account, starting from a limited responsiveness of the system to the unavailability of the service, hanged waiting for a mandatory packet, that the web-service architecture of GRAPPLE isn't able to serve.

6.3 SAKAI

6.3.1 Description of the LMS

Sakai is an Open Source research teaching and learning platform. It was originally developed under an Andrew Mellon Foundation grant by MIT, Stanford, Michigan and Indiana Universities to provide collaborative support for research groups as well as teaching and learning. The project has finished, and Sakai is now nurtured by the Sakai Foundation, a not for profit 501C organization registered in the US. The code base is maintained by about 50 core developers, whose time is donated from participating institutions. The software is in full production use at approximately 161 Higher Educational institutes world wide supporting communities ranging from a few 100 to 200,000 users.

The University of Cambridge has been running Sakai as its core VLE/VRE for 3 years now, supporting 5000 unique users per week.

Sakai is written entirely in Java and works as a web application serving.

The core bundle contains about 15 tools, but there are at least another 20 tools contributed by individuals and institutions.

6.3.2 Integration with Grapple ALE

Although Sakai is capable of providing fine grained event feeds of the impact of every action, only higher level events will be sent to the GRAPPLE ALE. This communication will be performed with one of the standard SOAP libraries to be determined once the precise dialect of SOAP has been specified to avoid interoperability issues.

Call back web services will be made available where more information is required from Sakai.

6.3.3 Limitations and Risks

Sakai event processing is fast and does not contain network callouts, consequently it is in the request cycle. Any latency in event processing will result in slower response at the user interface. Although it is perfectly feasible to make the event processing asynchronous, it is probably outside the scope of the GRAPPLE project. For this reason, there is a risk that the current event architecture within Sakai, and the speed of processing events on the ALE will result in slow performance.

6.4 IMC CLIX

6.4.1 Description of the LMS

CLIX® is the acronym for Corporate Learning and Information eXchange and is the learning management system of IMC, Germany. It is widely used in European enterprises (more than 100 companies are using CLIX as learning technology). It is a learning management system which has been conceived as a Java web application for which the user needs a web browser (client).

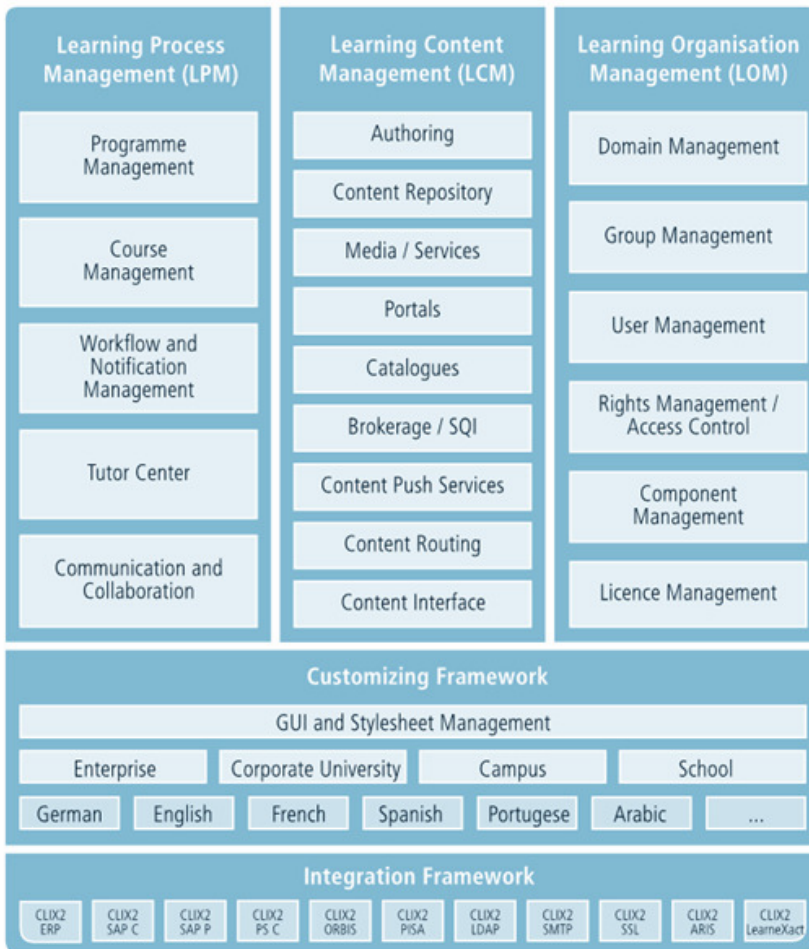


Figure 18: Detailed view on the learning management components of CLIX.

Figure 18 shows the CLIX architecture and its components. The CLIX core functions support learning and training processes, the administration of learning contents and the management of users, learning groups, roles and organisational domains. All functionalities are bundled in the components Learning Process Management, Learning Content Management and Learning Organisation Management.

Learning contents, such as web-based trainings, MS PowerPoint slides, tests and feedback sheets can be integrated into courses. The teaching plans and curricula within courses can be freely defined and structured by learning logics. Learning communities can be established by using tools such as chats and forums. All activities can be tracked and evaluated via a wide range of reporting options.

6.4.2 Integration with GRAPPLE ALE

CLIX is currently supporting the following standards that could be used (may be modified or enhanced) for integration purposes: Dublin Core, LOM, AICC, IMS QTI, and ADL SCORM.

User data can be imported from both CSV (Comma Separated Values) files and LDAP (Lightweight Directory Access Protocol) directories. A LDAP server can also be used as an authentication source.

In addition, CLIX® utilizes both JMS (Java Message Service) and Web services for further extensions, data exchange and communications with external components. A Web service can be provided in CLIX® by implementing a respective Java class which provides one or more public methods which, in turn, return the data expected by the consumer. This Java class in conjunction with an XML description file (WSDO file) becomes a Web service. Further, a Web service can be embedded in CLIX® as a server or client. CLIX uses



the Java implementation of the Apache Axis package (Apache eXtensible Interaction System). This implementation supports the standards SOAP 1.1, 1.2 and WSDL 1.1.

There are already some Web services implemented in CLIX that provide information requested by external applications. These Web service methods are:

- `getAllCourses()`: Returns all registered courses in CLIX
- `getCoursePermissions()`: Returns access rights of a specific course
- `getChats()`: Returns all chats assigned to a specified course
- `getClixId()`: Returns the internal identifier of a learning component
- `getCourses()`: Returns all courses available for a specified user
- `getMyCatalogs()`: Returns a list of catalog objects accessible for a specified user
- `getMyCourses()`: Returns all courses that are or completed booked by a specified user
- `getMyAppointments()`: Returns all dates of a specified user
- `getFAQs()`: Returns all Frequently Asked Questions (FAQs) components that are available for the specified user
- `getGlossaries()`: Returns all glossaries that are available for the specified user.

Furthermore, CLIX® supports Shibboleth that facilitates a user to log in once and gain access to the resources of multiple software systems without being prompted to log in again.

CLIX tracks high level events, such as the learner access of learning components, but does not provide an interface for external applications yet. But events could be provided by a Web service or sent to a Web service if necessary.

6.4.3 Limitations and risks

As CLIX uses the Java implementation of the Apache Axis package, deployed Web services only support SOAP 1.1, 1.2 and WSDL 1.1.

Further there could be a limited responsiveness of the system due to unavailable Web services or slow response of a Web service employed in the Event Bus in case of synchronous communication.

6.5 learn eXact

6.5.1 Description of the LMS

learn eXact is a Learning Content Management System that combines the administrative and management dimensions of a traditional LMS with the content creation and personalized assembly dimensions of a CMS. From this consideration it is possible to state the equation $LCMS = CMS + LMS$.

The **learn eXact platform** is a complete LCMS that allows creating, managing and delivering *e-learning* (Distance Learning based on the use of the Web) contents.

The **learn eXact** platform is completely based on international standard specifications (SCORM, IMS, AICC, etc.) for indexing, packaging, delivering and tracking *Learning Objects*.

In particular, these specifications come from the following institutions:

- **IMS** (www.imsproject.org), regarding learning content meta-data (*IMS Metadata*, used by SCORM), content resource aggregation and course organization (*IMS Content Packaging*, used by SCORM) and persons and groups data (*IMS Enterprise*);
- **ADL/SCORM** (www.adlnet.org), regarding both the content creation on the CMS and the content delivery on the LMS, including the tracking environment (*SCORM Run Time Environment*);
- **AICC** (www.aicc.org), just regarding the content creation on the CMS.

Besides, the **learn eXact** platform manages the information defined by the above mentioned specifications in their native form, i.e. describing and storing them on the Digital Repository through the XML (*eXtended Markup Language*) technology, which supports the current and future e-learning specifications and allows separating data structures from their rendering.

The **learn eXact** software is made out of three independent and interoperable modules:

- The **eXact Packager** is the Client module used to create contents, both courses (according to standard IMS SCORM, AICC formats or proprietary formats), and Learning Objects, i.e. simple or complex self-contained chunks of instruction, or assets that can be re-used in different courses. This module also allows indexing row assets (images, videos, etc.) using meta-data.
- The **eXact Lobster** (*Learning Objects Brokerage and StoraGE Repository*) is the (local or remote) database used by the **eXact Packager** module as (XML native) Digital Repository to store and retrieve resources, Learning Objects and courses described by means of metadata. It's based on the XML Tamino Database provided by Software AG (www.softwareag.com). Besides, this module is used to store users information, both personal information and tracking information about the student performance on courses.
- The **eXact Siter** allows delivery and tracing contents (Courses) stored into the eXact Lobster according to the SCORM standard; it also offers facilities for course management, users' enrollment, (default and single user) layout personalization and communication tools (chat, forum, guestbook, etc.) management. This module includes the **eXact Tracer** module, i.e. the engine (according to the SCORM specification) that traces the student performance on courses.

The **eXact Packager** and the **eXact Lobster** modules constitute the **learn eXact Learning Content Management System** (LCMS) which is used to support the publishing process.

The **eXact Lobster** and the **eXact Siter** modules constitute the **learn eXact Learning Management System** (LMS) which is used to support the delivery process.

Figure 19 illustrates the modular structure of the system:

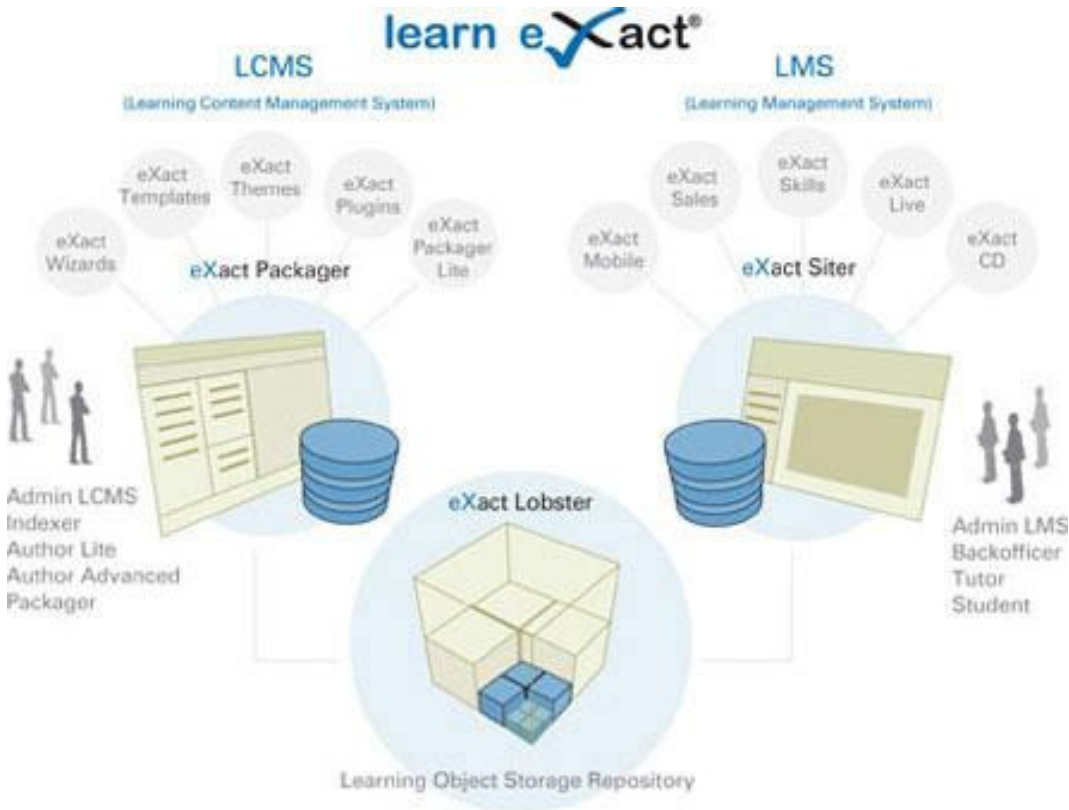


Figure 19: learn eXact system.

It provides course management and user enrollment functionalities as well as community tools for the end users.

With eXact Siter you can set up your Learning Portal and adapt it to your learning community by customizing its layout skin, creating thematic community environments and managing broadcasting events such as news and virtual meeting places.

Learning Management Processes in eXact Siter include the following main features:

- Users management and profiling (Learners, Tutors, HR Manager)
- Management of learning and assessment materials on delivery
- Management of classes (thematic learning environments)
- Events and calendar management (plan learning activities for single users or classes)
- Mobile and Location Based Learning (Windows Mobile™, Blackberry™, Symbian™)
- Skills management (plan and monitor learners' gap analysis and assign remediation plans)
- Taxonomy-based Content Personalization for Skills and Competencies
- eCommerce portal features (sell your courses online)
- User performance tracking (SCORM certified), reporting and statistics
- Fully customizable navigation menu
- Unicode Multilingual support.



D7.1a - Initial specification of the operational infrastructure (v1.0), 28-01-2008

Learning contents, such as web-based trainings, MS PowerPoint slides, self assessment tests can be integrated into training courses. Learning communities can be established by using tools such as chats and forums with proprietary mechanism. All activities can be tracked and evaluated via a wide range of reporting options.

Then learn eXact supports the following standards: IMS CP, IMS QTI, IMS SS, IMS ENTERPRISE, LOM, Dublin Core (mapped on LOM), and ADL SCORM.

6.5.2 Integration with GRAPPLE ALE

learn eXact uses capabilities of the framework .NET to connect to external tools.

6.5.3 Limitations and risks

Further there could be a limited responsiveness of the system due to unavailable Web services or slow response of a Web service employed in the Event Bus in case of synchronous communication.

References

1. T. Erl, *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, Prentice Hall, 2004
2. T. Scavo, S. Cantor, *Shibboleth Architecture – Technical Overview*, 8th June 2005
3. D1.3a, “Stand-alone” Adaptive Learning Environment - specification
4. D2.1a, “Definition of an appropriate User profile format”
5. D3.1a, Design specification of a DM definition tool
6. D3.2a, Definition of a concept relationship type tool
7. D3.3a, Design of a CAM definition tool
8. D4.5a, Design of interactive visualization of models and students data
9. D6.1a, Distributed user model platform and retrieval service – design
10. D10.1, Requirements specification by corporate users of LMS on integrated adaptive learning services.