A Grid Based IMS Learning Design Player: the ELeGI Case Study

Nicola CAPUANO [a, b, 1], Roberto IANNONE [a], Sergio MIRANDA [b, c] and Marcello ROSCIANO [a]
[a] *CRMPA, Centro di Ricerca in Matematica Pura ed Applicata, Italy*
[b] *DIIMA, University of Salerno, Italy*
[c] *MOMA, Modelli Matematici e Applicazioni, Italy*

Abstract. This paper illustrates the work done and the results achieved within the ELeGI project about the orchestration and the delivery of Learning Services lying in the GRID inside an IMS Learning Design (IMS-LD) Unit of Learning and running under an enhanced version of the CopperCore Player. The added value of GRID technologies for the creation and the execution of dynamic learning experiences is evidenced as well as the experimentation performed to overcome the original IMS-LD limitation on running services is presented. The aim of the ELeGI project is to promote and support a learning paradigm centred on the knowledge construction using experiential based and collaborative learning approaches in a contextualised, personalised and ubiquitous way through the definition and implementation of a service oriented Grid based software architecture.

Introduction

The aim of the ELeGI project is to promote and support a learning paradigm centred on knowledge construction using experiential based and collaborative learning approaches in a contextualised, personalised and ubiquitous way through the definition and implementation of a service oriented Grid based software architecture.

In the context of the ELeGI project a new approach to the learning design web service integration has been studied. It exploits the CopperCore architecture (engine and player) in a distributed learning environment. This paper describes the realized framework able to generalise the web service integration inside an IMS LD Player.

In Section 1 the ELeGI project and its goals are briefly described. Section 2 introduces the IMS Learning Design specification and the actual engine and player implementation. CopperCore engine is described as well as the SLeD project, and the actual limitations related to the web service integration issues. Eventually, Section 3 details the realised approach.

[1] Corresponding Author: CRMPA, Centro di Ricerca in Matematica Pura ed Applicata, Università di Salerno, Via Ponte Don Melillo, 84084 Fisciano (SA), Italy; E-mail: ncapuano@unisa.it.

1. The ELeGI project

1.1. Objectives

The European Learning Grid Infrastructure (ELeGI) project [1], a EU-funded Integrated Project, aims at promoting and supporting the adoption of a learning paradigm focused on the construction and sharing of knowledge in high dynamic contexts using experiential based and collaborative learning approaches in a ubiquitous, contextualised, and personalised way and taking into account informal learning aspects as well. To achieve these goals, ELeGI defines a clear strategy, formalized through the definition of models (supporting formal and informal learning scenarios), methodologies and technologies, enabling to overcome the drawbacks of traditional e-Learning solutions and to advance the effective use of Technology Enhanced Learning (TEL).

1.2. The ELeGI Learning Model

The Learning Model defined in the ELeGI project and fully described in [2] is conceived in such a way to represent different pedagogical approaches and to allow the automatic building and delivery of adaptive Unit of Learning (UoL) that can dynamically change during the learning process according to the learner's pedagogical needs and preferences. The ELeGI Learning Model produces an operational process so to create and deliver a UoL by using the structures defined by the three underlying specific models: the Knowledge Model, the Learner Model and the Didactic Model. The overall operational process can be divided into three processes, each one addressing a specific phase: Knowledge building process, UoL building process and UoL delivery process.

Unlike the traditional approaches to the creation of UoLs and/or learning courses, the exploitation of the learning model offers several advantages. Actually, examining current tools (IMS-LD tools, LAMS, etc.), a UoL can be considered as a static and monolithic block, since once created, it is rather difficult to change or modify its inner resources and/or to add/remove services and resources at run-time. Moreover, the traditional approaches to create a UoL typically rely on expertises of the institutional designer only, and have no capability to reuse existing blocks. Conversely, exploiting the Grid technology and the virtualisation mechanism (by which each resource is virtualised as a service), the ELeGI approach allows both to add dynamicity during the UoL delivery, enabling the automatic search and late binding of resources and services, and to reuse already developed building blocks (e.g. Ontologies, LEM, Didactic Methods) during the UoL building. Thus, the benefits of the ELeGI approach, from the reusability and pedagogical viewpoints, appear to be noticeable.

2. A Learning Design implementation

2.1. IMS Learning Design

The IMS Learning Design is a specification used to describe learning design scenarios. It can describe a wide variety of pedagogical models, or approaches to learning,

including group work and collaborative learning. It does not define individual pedagogical models, it provides instead a high level language, or meta-model, useful to describe many different models. The language outlines how people perform activities using resources (including contents and services), and how these three components are coordinated into a learning flow.

The IMS Learning Design illustrates how a learning design scenario unfolds making an analogy with a theatrical play. Just as a play can be staged by different actors, in different theatres/places and changing props, so learning design scenarios can be executed again by different learners and tutors, on different systems, with alternative learning resources or tools:

- The play is presented in a series of acts, in which roles are played by those actors taking part in it, for instance: learner, tutor, mentor, and so on.
- People playing the roles undertake a series of activities within an act. For a learner these might include discussing with classmates about the merit of a piece of source material. A tutor's activity could be to remark on their conclusions.
- Each role is presented together with its own learning objects (LO) and services (e.g. communication tools) within an activity.
- An act is complete after all the activities of a specified role, or roles, are finished. Alternatively, a time limit may be set, which determines an act to be completed.
- As soon as an act is completed, the next one starts. The play finishes when all the acts are complete, the learning design scenario finishes when all the plays are complete.

The IMS Learning Design specifications are structured into three levels:

- *Level A* includes activities, roles and environments. Activities (learning activities or support activities) can be grouped into activity structures and executed into specific environments. An environment is composed of learning objects and services provided to the users during the activity execution. Users are classified into roles (learners, teachers, tutors, etc…). Learning objects are educational contents by which learners acquire knowledge and services are functionalities invoked during the learning process in order to communicate with tutors or other learners.
- Level B *adds properties (storing information about a single person or a group) and conditions (setting constraints upon the flow of activities) to the first level.*
- *Level C* adds notifications (such a mechanism handling messages between users) to the framework.

As observed in [3], even though IMS-LD presents a mechanism to personalize the learning experience at run-time using properties and conditions at Level B, the instructional designer has to provide a fixed set of learning objects and services in which to select contents and tools for final users of the learning experience. So learning objects nd services are statically bound to the learning design scenario.

2.2. CopperCore

CopperCore is the world's first open source Learning Design engine capable to process all three levels of IMS Learning Design. Released from Open Universiteit Nederland (OUNL) it is not designed be used as a stand-alone learning environment but to be integrated in a service oriented framework consisting of different services that are combined to create a complete e-learning system. CopperCore architecture provides three application program interfaces (APIs). The first provides access to validation; the second to services; and the third provides the presentation aspects.

All the APIs are exposed using XML and can be called or manipulated externally to provide file management and student management. This allows the separation of content and services from the engine itself. The CopperCore system also provides a LD player implemented as XSL style sheets applied to the transformed learning design. The core implementation uses Java to provide the combining and integration code.

2.3. Service delivery inside the SLeD project

The SLeD project (Service Based Learning Design Player) [5] was the result of a collaboration between the UK Open University (UKOU) and the OUNL and was funded as part of the Framework and Tools strand of the JISC e-Learning Programme, which is constructed around the concept of a service oriented architecture [6]. The project was built on the CopperCore engine which validates Learning Design packages to check if they conform to the specification, and if not, indicates the problem areas.

The objectives of the project were focuses on the extension of the available tools to the Learning Design community and were also intended to further develop the way a Learning Design approach might be practically realized within an institution. One of the aim of the project was to extend and formalise the approach for integrating services while maintaining the architectural integrity of the system. The Sled architecture is shown in the figure 1.

The CCSI module is a layer that sits between the Sled (or other learning design) player and the other LD services (for example the CopperCore engine, forums, search etc). It provides a single point of contact thank to which the player may access all the services, and also coordinates the communication between the services. For example, with the QTI service, the CopperCore LD engine also needs to know the results from the users' test, so when the user submits his/her answers, CCSI will pass the submission received from the user to the QTI engine, and then return the test results back to the user and to the CopperCore engine. Communications between player and services work due to the fact that the player is connected to CCSI by a set of predefined methods (for each service used), the CCSI module then transforms/adapts this method call to that actually expected by the service provider.

CCSI also handles the transformation of the information back from the service provider to the player. In this way the player always knows the method to call and that one it should expect back - no matter who is providing the actual service, it is a task of the CCSI layer to ensure that the transformations take place correctly. For each type of service (e.g. LD engine, forum, search) there is an adapter class which provides the actual implementation and tranformations for the player to connect to these services. Any parameters required by the adapters (for example URLs to web services) are contained in a configuration file.

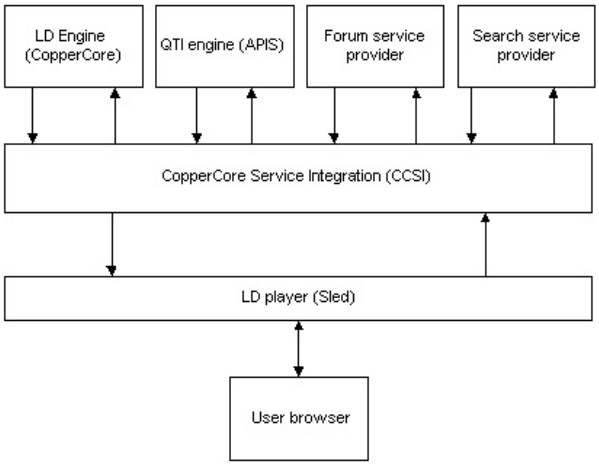

Figure 1. SLeD Architecture

2.4. Actual limitations

Even though the SLeD project has considerably improved the service integration into IMS Learning Design player some drawback already persists.

In order to switch between service providers for which an adapter class is available, e.g. to switch between a Google search provider and a MSN search provider, it is necessary to manually modify an xml configuration file in the CCRT folder of the CopperCore installation and then to restart CopperCore for the changes to take effect. Furthermore it is possible only a single connection to a provider for a particular service at a time. For example it could not have a Unit of Learning which uses a Google search engine and another Unit of Learning configured to use MSN search.

Adding a new service type is more complicated than switching service providers. The steps to carry out are the following:

- To think to the generic functions for the new service
- To code the adapter classes to connect up to the actual service provider.
- To update the player so that it will connect to the new service type.
- To create the interface the authors could use to write unit of learning that embeds the new service.

3. Service delivery inside ELeGI IMS-LD Player

The ELeGI Software Architecture is presented in Figure 2. The Grid layer provides a set of Infrastructure Services and other services to create and manage a Virtual Organisation (VO) [8]. The Learning layer is mainly devoted to the execution of the processes related to the Learning Model.

The Learning Services sub-layer, provides services and tools to support the execution of the three processes of the Learning Model including the management of

Ontologies, the Learner Model, and the Didactic Model. The Personalization subsystem aims at dynamically adapting and delivering educational contents and services, matching learner's needs and preferences according to his/her profile. The Learning Experience Management subsystem allows to access and manage courses, modules, and other learning experiences (e.g. allocating student, staff, etc.), while Contents & Services Orchestration subsystem deals with the execution of the UoL, that are described using the IMS-LD [4].

The Application Layer uses the services provided by the underlying layers, or their composition, to implement applications in the e-learning domain. This layer includes a key component of the ELeGI software architecture, IWT Grid Aware (IWT-GA) [9] Portal that, according to the research on Grid portals, is designed exploiting the Web Services for Remote Portlets (WSRP) specifications [10].

IWT-GA is the Grid version of Intelligent Web Teacher (IWT) learning platform [11], it adopts the concept of portlets as a way to design user-centric portals that can be dynamically adapted to the context.

IWT-GA holds an IMS-LD compliant engine and Player, a modified version of Coppercore [12] that exploits the Contents&Services Orchestration subsystem services of ELeGI, that is the subject of the following discussion.

3.1. Service implementation

To have a service running inside the EleGI LD Player is quite simple, indeed it only needs to follow a few implementation specifications for the service development and its deployment in the GRID. There is a loosely-coupled connection between the service and the UoL or the Player that will run it, indeed no specific reference to the service, or particular installation inside the player must be done, the only requirement is to specify inside the IMS-LD learning activity the metadata of the required service.

In order to implement a new service or to extend an existing one, and made it compliant with and fully executable within the IMS-LD Player here referenced to, some APIs have been developed and made available.

The services must be a WSRF compliant GRID service (the WSRF.Net framework which the University of Virginia Grid Computing Group has adopted) [16] in order to maintain its state during the various phases of the execution.

By simply extending a class provided by the mentioned APIs, the service inherits the state management logics and the methods useful to communicate with the LD Player. A typical communication between the two parts can be summarized by two classes of methods. In short, the Player uses a Setter method to pass to the service its initialisation parameters and two methods to know the references of the service delivery portlet to render. The service instead can use the Getter method to read the parameters passed. All the methods can be overriden in the service if the default implementation behavior is unsuitable, otherwise the simple class extension will be sufficient.

It is important to assign in the code the right WSDL base name to the GRID service, since the LD Player in order to communicate with every service needs to use a general service proxy, therefore the methods to be reachable on the service must have a fixed namespace.

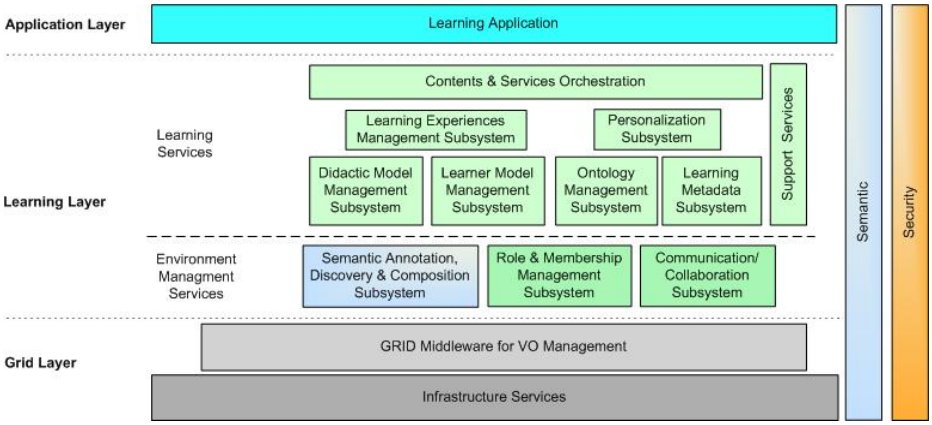

Figure 2. The ELeGI Software Architecture

3.2. Services discovery and binding

The web service integration approach described in this paper is slightly different from that adopted in SLeD architecture. The idea is to use a special xml configuration file that follows the xml schema depicted in the figure 3.

The SLA element follows a schema that is an extension of the Web Service Level Agreements (WSLA) standard [13]. WSLA specifications are agreements between a service provider and a customer defining the obligations/duties of the parties involved in it. Primarily, this is the obligation of a service provider to perform a service according to agreed-upon guarantees for IT-level service parameters (such as availability, response time and throughput) for Web Services, also specifying the measures to be taken in case of deviation and failure and to meet the asserted service guarantees (for example, a notification of the service customer).

This file represents a description of the service which includes the parameters it needs, specified by "param" element in the schema, and the discovery method used to retrieve the service. There are two implemented retrieving methods:

- Static address specification. Simply the service address specified in the "address" attribute of the root element is used to instantiate it;
- Dynamic address specification. This method is adopted if the "address" attribute is not specified. Then the "SLA" element is considered. This element contains all the information needed to retrieve a service registered in a UDDI repository.

The dynamic address specification needs that the desired service is registered using GRASP, a Grid middleware developed in the frame of the homonymous FP5 project [7], and successively re-factored in order to be WSRF compliant by exploiting the WSRF.NET implementation of the University of Virginia Grid Computing Group.

The service description file can be inserted into the desired UoL as a learning object within the learning activity where the service should be used. The ELeGI LD Player (a modified version of the CopperCore Player) is used to parse the description

file. When in a UoL play the user reaches a learning activity that entails a service support, the LD Player parses the file to search for retrieving the method to exploit.

If the static address specification method is to be used, the LD Player binds the referenced service to the generic service proxy, otherwise it tries to locate and instantiate the service using the GRASP middleware functionalities.

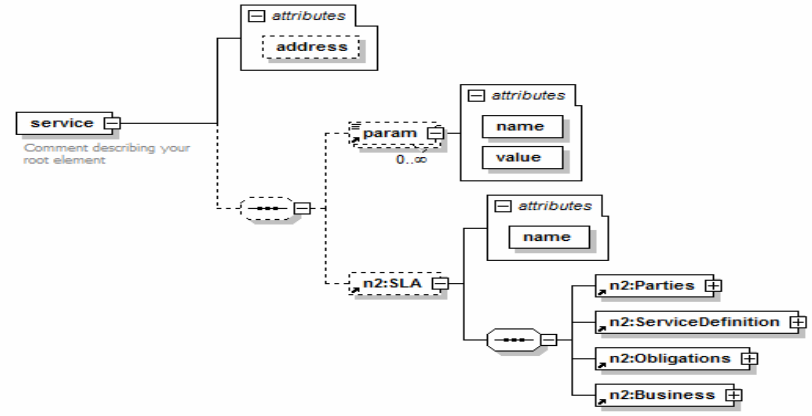

Figure 3. service description file XML schema

3.3. Services delivery

The main characteristic denoting the IMS-LD Player of the ELeGi project is the ability to discover and run services at run-time (i.e. which are not known prior to the execution of the UoL). Since those services are not installed under the same machine of the IWT-GA Portal server and/or of the Player, but they can be deployed in different places of the VO, the service GUIs are rendered in the Player as portlets through the WSRP.Net implementation (developed in the context of the ELeGI project).

In Figure 4 are outlined the steps executed to render a portlet when a service is delivered in the LD Player.

After that the service is instantiated, the LD Player reads the parameters specified in the service description file and passes them to the service in order to initialize it with all the values needed during the execution. Then the information required to render the portlet are returned while a component called WSRP Consumer, in the LD Player, is responsible to communicate with a WSRP Producer, referenced in the *GetWSRPProducerRef()* method, in order to get the portlet markup and to render it. The portlet communicates directly with the service to execute the business logic and expose its GUI.

In order to illustrate the above discussion, the “Matchmaking service”, a support service implemented in the EleGI project scenarios, will be used as an example. This service provides a person search support for users that during a learning activity need to look for a tutor or an expert with regard to a specific argument explained during the learning experience.

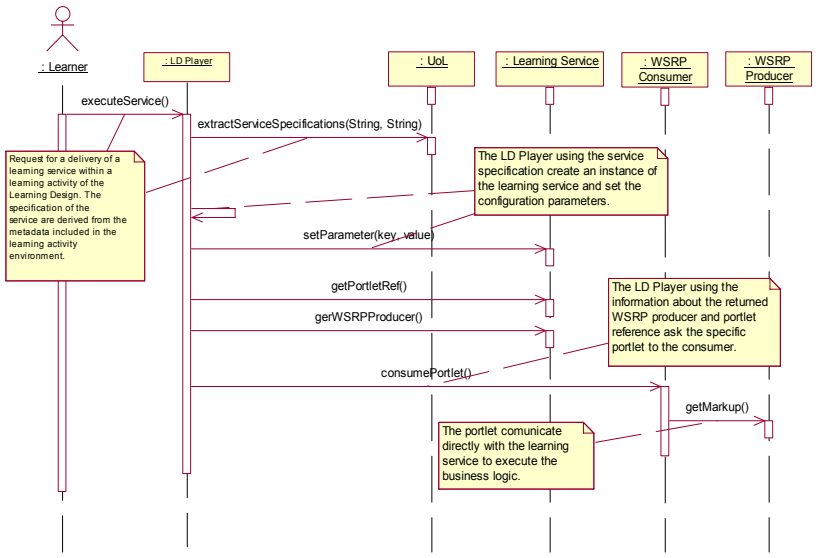

Figure 4. service delivery sequence diagram

Initially, a service description file embedded into the unit of learning (see example in figure 5), is read. Since the address attribute is empty, the dynamic address specification will be used so that all the information needed to retrieve the service URL could be extracted from the SLA element (see the WSLA standard for a better understanding [13]). Than some fixed parameters will be extracted from the "params" section of the service description file; in the example the parameters of interest are "filtered_search" where a value of "true" indicates a search not limited to the people engaged in the same unit of learning of the user requesting the search. and "type_of_role" where a value of "all" specifies to search for learner and/or teacher roles.

Finally the GUI portlet (depicted in the figure 6) will be rendered by the client. It allows to select the concepts and the knowledge level the searched people need to master.

4. Conclusions

In this paper has been presented an approach to integrate web service delivery inside a IMS-LD player. Besides, a modified version of the CopperCore player has been developed in the context of the ELeGI project in order to deliver, within a IMS-LD unit of learning, some dynamically discovered web services exploiting the Grid infrastructure.

The solution proposed seems to be flexible enough to theoretically allow also the delivery of services implemented through the Java language. Indeed the service should only implement an interface (defined in the APIs) and expose it via Web Service. Furthermore, the WSRP Net Consumer implementation has demonstrated in the interoperability tests to allow to consume portlets produced by Java implementations of

the WSRP specification. In the next future the experimentation work will proceed towards this direction.

```
<?xml version="1.0" encoding="utf-8"?>
<service xmlns="http://www.elegi.org" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.elegi.org LDPS.xsd" address="">
    <params>
        <param name="type_of_role" value="all"/>
        <param name="filtered_search" value="false"/>
        <param name="domain" value="Mathematics"/>
        <param name="concepts" value="conc1,conc2,conc3"/>
    </params>
    <SLA xmlns="http://www.ibm.com/wsla" name="MMS_SLA">
        <Parties>
            <ServiceProvider>
                <Identifier>Identifier</Identifier>
                <Address>Street</Address>
            </ServiceProvider>
            <ServiceCustomer>
                <Name>Name</Name>
                <Identifier>Identifier</Identifier>
                <Address>Street</Address>
            </ServiceCustomer>
        </Parties>
        <ServiceDefinition name="MatchMakingService">...
        <Obligations>
            <ServiceLevelObjective name="GlobalResponseTimeObjective">
                <Obliged>provider</Obliged>
                <Validity>...
                <Expression>
                    <Implies>
                        <Expression>
                            <Predicate xsi:type="Less">
                                <SLAParameter>globalOperations:CPUUse</SLAParameter>
                                <Value>99</Value>
                            </Predicate>
                        </Expression>
                        <Expression>
                            <Predicate xsi:type="Less">
                                <SLAParameter>globalOperations:CPUUse2</SLAParameter>
                                <Value>100</Value>
                            </Predicate>
                        </Expression>
                    </Implies>
                </Expression>
            </ServiceLevelObjective>
            <ActionGuarantee name="g1">...
        </Obligations>
        <Business>...
    </SLA>
</service>
```

Figure 5. service description file for the Matchmaking service

References

[1] M. Gaeta, P. Ritrovato, S. Salerno. Making e-Learning a Service Oriented Utility: The European Learning Grid Infrastructure Project. chapter for the book: Towards the Learning GRID: advances in Human Learning Services, IOS Press, ISBN: 1-58603-534-7, pp. 63-75, 2005

[2] G. Albano, M. Gaeta, S. Salerno. E-learning: a model and process proposal. To appear in International Journal of Knowledge and Learning, 2005.

[3] N. Capuano, M.Gaeta, R. Lannone, F. Orciuoli. Learning Design and run-time resource binding in a distributed e-learning environment, in Proc. of the 1st International Kaleidoscope Learning Grid SIG Workshop on Distributed e-Learning Environments, Vico Equense (Napoli), Italy, BCS Pub, March 2005.

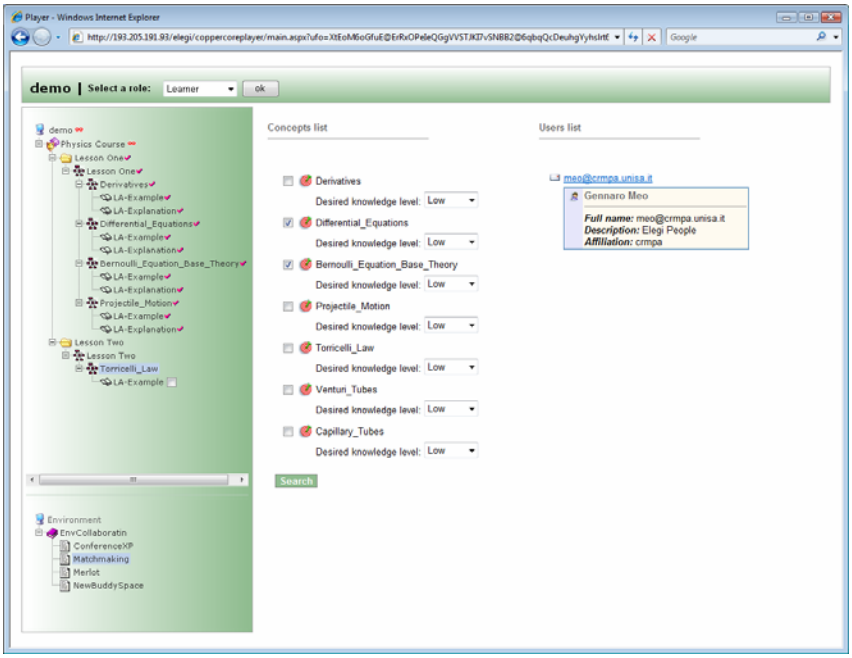

Figure 6. Matchmaking portlet inside the LD Player

[4] IMS Global Learning Consortium, IMS Learning Design v1.0 Final Specification, available at http://www.imsglobal.org/learningdesign/index.cfm, 2003.
[5] Service Based Learning Design Player, available at http://sled.open.ac.uk/web/.
[6] The JISC e-Learning programme, available at http://www.jisc.ac.uk/whatwedo/themes/elearning/programme_elearning.aspx
[7] T. Dimitrakos, M. Gaeta, G. Laria, et al.. An Emerging Architecture Enabling Grid Based Application Service Provision. Proc. of 7th IEEE Int. Conf. EDOC, Brisbane, Australia, pp. 240-250, 2003.
[8] Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications, vol. 15, no. 3, pp. 200-222, 2001.
[9] N. Capuano, A. Gaeta, G. Laria, F. Orciuoli, P. Ritrovato. How To Use GRID Technology for Building Next Generation Learning Environments. chapter for the book: Towards the Learning GRID: advances in Human Learning Services, ISBN: 1-58603-534-7, pp 182-192, 2005.
[10] T. Schaeck, R. Thompson: Web Services for Remote Portlets (WSRP) Whitepaper, 28 may 2003. Available at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp
[11] Intelligent Web Teacher, available at http://www.didatticaadistanza.com/.
[12] CopperCore The IMS Learning Design Engine, available at http://coppercore.sourceforge.net/
[13] Web Service Level Agreements (WSLA) Project, available at http://www.research.ibm.com/wsla/
[14] M. Weller, A. Little, P. McAndrew, W. Woods. Learning Design, generic service descriptions and universal acid. Educational Technology & Society, 9 (1), 138-145, 2006.
[15] P. McAndrew, W.I.S. Woods, A. Little, M.J. Weller, Rob Koper, Hubert Vogten. Implementing Learning Design to support web-based learning, available at http://ausweb.scu.edu.au/aw04/papers/refereed/mcandrew/
[16] WSRF.NET, available at http://www.cs.virginia.edu/~gsw2c/wsrf.net.html