

Exploiting agents in e-learning and skills management context¹

Alfredo Garro^{a,*}, Luigi Palopoli^a and Francesco Ricca^b

^a D.E.I.S., Università della Calabria, Via P. Bucci, 87030 Rende (CS), Italy
E-mail: {garro,palopoli}@deis.unical.it

^b Dipartimento di Matematica, Università della Calabria, Via P. Bucci, 87030 Rende (CS), Italy
E-mail: ricca@mat.unical.it

Abstract. Nowadays, it is quite agreed that organizations gain limited advantages in adopting e-learning platforms that only provide educational contents. An advantageous e-learning platform should have instead the capability to help enrich, share and circulate organization knowledge, thus contributing to making the organization dynamic and flexible. In this paper MASEL, a Multi-Agent System for E-Learning and Skill Management is described. MASEL performs the following tasks: (i) supports Chief Learning Officers in defining roles, associated competencies and required knowledge level; (ii) manages the *skill map* of the organization; (iii) evaluates human resources competence gaps; (iv) supports employees in filling the competence gaps related to their roles; (v) creates personalized learning paths according to feedbacks that users provide to optimize the acquisition of required competencies; (vi) assists Chief Learning Officers in selecting the most appropriate employee for a given role; (vii) assists a Project Manager in building teamwork. A prototype tool implementing MASEL using JADE (*Java Agent DEvelopment Framework*) was developed. The reasoning capability of MASEL agents involved in the learning paths building process and in the team building process is implemented using DLV, a disjunctive logic programming system.

Keywords: Software agents, Enterprise Knowledge Management, e-learning, skill management

1. Introduction

E-learning can be described as supporting a learning experience by developing and applying Information & Communication Technology [43]. While the advantages of e-learning have been for years characterized simply in terms of cost effectiveness gained (both in terms of time and space), it is nowadays widely recognized that its potentialities go far beyond this, involving issues like diversification of learning paths and general business competitive advantage [26,27,31,32].

Because of the dynamics of the market, often organizations cannot program in the medium-long term, but need to work in a project-shaped, short-to-medium term perspective. When an organization is to carry out a project, new competencies are typically to be acquired, which are frequently expensive and hard to find

in the external market (the *skill shortage* problem) and it is often the case that such competencies must be found (or constructed) internally. An appropriate and properly used e-learning platform thus becomes an important component of Enterprise Knowledge Management [9,33]. Given a project specification, the platform should be able to suggest a project team, to measure human resources competence gaps and to contribute to reduce them by creating personalized learning paths. Moreover, the platform should be able to dynamically readapt learning paths according to feedbacks that users provide to optimize the acquisition of required competencies.

The implementation of an e-learning platform able to carry out the activities listed above requires technologies enabling communication between distributed components which should be able to:

- perform sequences of complex operations based on the received messages;
- perform complex tasks possibly using automatic reasoning techniques;
- be adaptive to user needs.

¹ A preliminary and partial version of this paper appears in *Proc. of the 3rd International Symposium on Multi-Agent Systems, Large Complex Systems, and E-Businesses (MALCEB'02)*, Erfurt, Germany, October 2002.

*Corresponding author: Alfredo Garro, D.E.I.S., Università della Calabria, Via P. Bucci, 87030 Rende (CS), Italy.

Agent technologies are conceived to support the features listed above. In fact, agents can perform sequences of complex operations based on messages they receive, their own internal beliefs and their overall goals and objectives. Furthermore, agents are expected to be proactive, interactive, adaptive and to have the ability to reason [24]. The aim of this work is to evaluate the contribution that agent technologies can offer in the e-learning and skills management context. To this end a Multi-Agent System for E-Learning and Skills Management (*MASEL*) was designed and prototyped, which is illustrated below.

In a few words, *MASEL* performs the following tasks: (i) supports Chief Learning Officers in defining roles, associated competencies and required knowledge level; (ii) manages the *skill map*² of the organization; (iii) evaluates human resources competence gaps; (iv) supports employees in filling their competence gaps as related to their roles; (v) enriches a given courseware or creates personalized learning paths according to feedbacks that users provide to optimize the acquisition of required competencies; (vi) assists Chief Learning Officers in choosing the most appropriate employee for a given role; (vii) assists a Project Manager in building teamwork.

The rest of this paper is organized as follows: in Section 2 the most largely adopted organization paradigm for educational contents is presented; in Section 3 *MASEL* is described in detail; in Section 4 related literature and a comparison among *MASEL* and various related systems are presented; finally, in Section 5, conclusions are drawn and some future activities being dealt with are presented.

2. Organization of educational contents for e-learning

Nowadays in the context of e-learning it is widely accepted that educational contents should be looked at as organized in relatively small independent units, called Learning Objects (LOs), which can be combined to create personalized learning paths.

However, there is no universally accepted formal definition of LOs in the literature. One of the most commonly used is that from IEEE - LSTC/LOM:³

²A skill map stores information about the role of each employee in the organization, the competencies required for that role and the current level of employee competencies.

³Institute of Electrical and Electronics Engineers – Learning Technology Standards Committee/Learning Objects Metadata Working Group.

“Learning Objects are defined here as any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning” [19].

Generally speaking, there is a common agreement that LOs have to feature the following properties [46]:

- atomicity. An LO is atomic in the sense that it is a (self)consistent *piece of knowledge*;
- reusability. An LO can be used (reused) in building various learning paths. In other words an LO can be shared by multiple learning-paths;
- repurposability. The ability to extract portions of a Learning Object and adapt them to new learning contexts;
- availability. An LO can be accessed any time, and everywhere, according to user needs;
- granularity. This property is defined by IEEE LSTC/LOM as “the functional size of the resource”. For example a “Textual Handbook” (.doc or .pdf) has low granularity, instead a “Hypermedia Guide” (.html) has high granularity;⁴
- inter-operability. A user should be able to use and put together Learning Objects developed by different people and organizations to support exchanging, reusing and knowledge sharing.

To achieve appropriate management of a Learning Object database, it is necessary to describe its content in an efficient and effective way. In other words, it is necessary to acquire a meta-knowledge that allows learning objects (documents, slides, simulations, roles, questionnaires, pre-recorded lessons, classroom lessons, ...) to be classified and their relationships with respect to objectives, topics, media, and so on. These are called Learning Object Metadata, LOM for short. In order to allow exchanging, reusing and sharing Learning Objects, it is required to express meta-knowledge using standard formats and protocols. In fact, standardization of Learning Objects description is one important goal for the e-learning community. In this respect, reference can be made to the following standards/organizations: LSTC (Learning Technology Standard Committee), ADL (Advanced Distributed Learning), ARIADNE (Alliance of Remote Instructional, Authoring and Distribution Network for Europe), IMS (Instructional Management System), DUBLIN, ESCOT (Education Software Component Of Tomorrow).

⁴An LO can be also seen as a collection of other LOs. If a LO consists of a large number of other LOs its granularity is high.

In particular, the authoritative organization IMS proposes to describe Learning Objects through XML documents validated with respect to *XML schemas* established by a standard [20]. This standard has been fully defined and most of the commercial e-learning platforms support it.

The availability of a set of metadata describing LO characteristics opens a relevant problem, which is to aggregate Learning Objects in order to obtain a courseware [46]. E-learning platforms feature tools that allow the user to search, manipulate, and aggregate Learning Objects manually. The possibility of automatizing the Learning Object aggregation [19] by using artificial intelligence techniques is considered an important goal of the next generation of e-learning solutions. Such a learning path building problem can be formalized as follows:

*Building a courseware (learning path) starting from a database of "Learning Objects" able to fill user competence gaps, in order for him to achieve the needed skill.*⁵

This task is very complex and its definition has many degrees of freedom. In the literature there is no universally accepted pedagogical methodology for solving this problem. An additional problem is that no instructional design information is included in the current version of the IEEE-LSTC/LOM standard [19]. This fact suggests that the systems based on this standard are unable to support a fully automated instructional development [46]. However, pragmatic (semi-automatic) approaches to this problem are available in literature (see [16]).

3. Engineering a multi-agent system for e-learning and skills management

In this section the engineering process followed to develop MASEL is described. This process consists in five steps: system requirements analysis, definition of the multi-agent system meta-model, system analysis and design, system implementation, and system testing and evaluation. The organization of this section follows that structure. First of all, the system requirements are discussed and the learning path building task (a core feature of MASEL) is described in detail. Then, a multi-agent system meta-model, defined on the basis

of the domain and requirements characteristics, is described. The defined meta-model represents the theoretical guideline followed during the analysis and design phases. The design of MASEL is then described by focusing on the description of agents services and cooperation protocols. Eventually, a prototype implementation of MASEL is presented and the related testing and evaluation activities are reported.

3.1. System requirements

Within the Skills Management context, the following main tasks need to be dealt with:

1. Singling out student learning objectives and evaluating his competence gaps.
2. Using a database of "Learning Objects" for building the courseware able to fill such competence gaps (*Learning Path Building*).
3. Checking the student improvements and integrating and (re)adapting the courseware content and presentation structure.
4. Creating a bridge between single user learning objectives and general organization learning objectives.
5. Managing the *skill map* of the organization and updating it according to the student learning improvements.

Besides the above basic tasks, it is also interesting to:

1. Select the most appropriate employee for a given role.
2. Select the most appropriate employees for carrying out a specific project (*Team Building problem*).

3.2. The MAS meta-model

On the basis of the requirements outlined in Section 3.1, a suitable MAS meta-model was defined in order to drive the analysis and design of the MAS.

The defined MAS meta-model is shown in Fig. 1 using an UML [44] class diagram. Due to the central role played in MASEL by the information exchange and representation, the *Ontology* meta-model elements constitute, in fact, an important part of the defined MAS meta-model. Moreover, due to the complexity of the application domain, a detailed specification of the *Services* provided by each agent is required. More specifically, an instance of the MAS meta-model is a MAS model consisting of a set of specific AGENTTYPES. An AGENTTYPE, which is

⁵A learning path should be personalized, according to the specificity (previous knowledge) of the individual user to be trained.

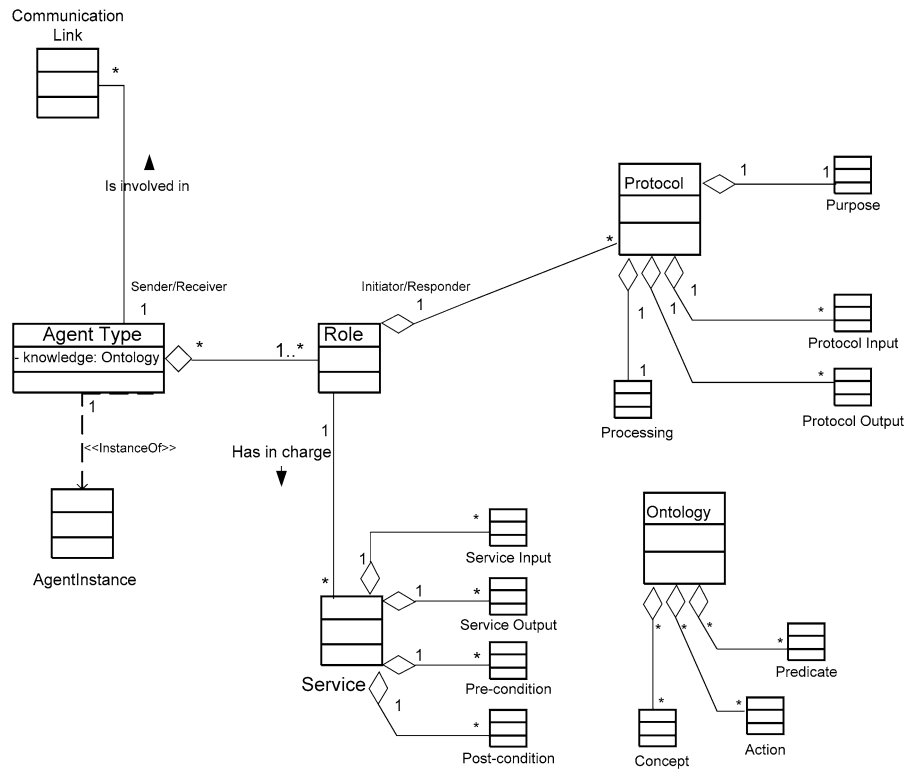


Fig. 1. The defined MAS meta-model.

involved in one or more COMMUNICATIONLINKS, is composed of a set of ROLES. A ROLE has in charge one or more SERVICES and participate (as INITIATOR or RESPONDER) in one or more Interaction PROTOCOLS [49]. A SERVICE is characterized by the following meta-model elements: INPUTS, OUTPUTS, PRE-CONDITIONS, and POST-CONDITIONS (which represent constraints on services). A PROTOCOL consists of the following meta-model elements: a PURPOSE describing the nature of the interaction; PROTOCOL INPUTS representing the information used by the protocol INITIATOR while enacting the protocol; PROTOCOL OUTPUTS representing the information supplied by/to the protocol RESPONDER during the course of the interaction; PROCESSING describing any processing the protocol INITIATOR performs during the course of the interaction.

Each AGENTTYPE handles an Ontology consisting of CONCEPTS, PREDICATES and ACTIONS [5].

It is worth noting that a MAS model, which is an instance of the defined MAS meta-model, consists of a set of specific AGENTTYPES; an instance of such MAS model is a *running* MAS consisting of a set of AGENTINSTANCES, one or more agent instances for each AGENTTYPE.

3.3. System analysis and design

In this section the *MASEL* multi-agent system is fully specified.

MASEL consists of eight AGENTTYPES, namely, Chief Learning Officer assistant agent (CLO), Skills Manager Agent (SMA), Student Assistant Agent (SAA), Learning Paths Agent (LPA), Content Agent (COA), Chief Content Officer assistant agent (CCO), Project Manager assistant Agent (PMA), User Profile Agent (UPA). *MASEL* agents are FIPA compliant [15], and able to operate, at least, in e-learning contexts based on the LOM paradigm according to the IMS standards [20].

Figure 2 shows the architecture of a generic *MASEL* multi-agent system. Each box represents a *MASEL* Agent and each arrow between two boxes indicates cooperation between the two associated agents. Intuitively, a specific assistant agent is associated with each user (according to the user typology: e.g., Student, Project Manager, Chief Learning Officer, Chief Content Officer) and system administration features and complex task execution are demanded to specialized agents (LPA, UPA, SMA, COA).

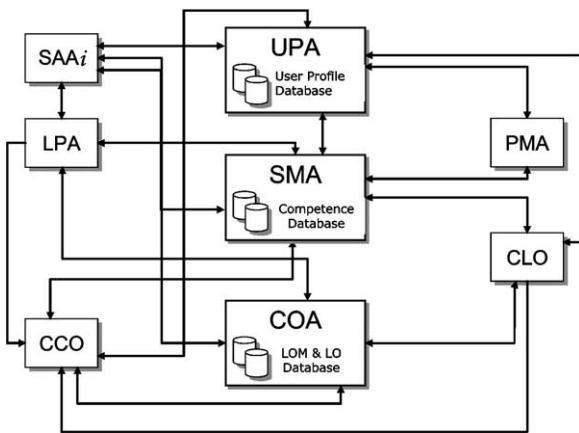


Fig. 2. MASEL architecture.

The following sections contain the formal specification of *MASEL* agents. In particular, for each agent the descriptions of the provided services and ontologies are given according with the agent meta-model described in Section 3.2. Interaction protocols specifying agent interactions are represented using AUML [1].

It's worth noting that the XML language is exploited both for representing and handling agents' ontologies as well as for managing data exchange among them. In particular, the agent ontologies are stored as XML documents; as a consequence, they are versatile and easy to exchange. In spite of this simplicity, the information representation rules embodied in XML are powerful enough to allow a sophisticated information management. Nowadays, a large number of XML-based formalisms exist for handling ontologies (e.g., SHOE [41], XOL [34], OML [35], RDF and RDF Schema [37], DAML+OIL [6] and OWL [45]). However, in *MASEL* it was opted to define and handle a pure XML document, validated against a well defined DTD or XML Schema, since this turns out to be sufficient for *MASEL* system requirements and avoids the need to handle verbose RDF(S) or DAML+OIL documents.

3.3.1. The Skills Manager Agent (SMA)

Provided Services. An *SMA* manages the skill map *SM* of the organization. *SM* stores the information, provided by a *CLO*, about roles and associated competencies. Moreover, for each employee Em_i , *SMA* manages the information about Em_i role and Em_i current competence levels. *SMA* provides the following services:

```
<!ELEMENT SkillManagerOntology (SkillMap)>
<!ELEMENT SkillMap (Role*, Competence*,
Employee*)>
<!ELEMENT Role (RequiredCompetence+)>
  <!ATTLIST Role
    Identifier ID #REQUIRED
    Name CDATA #REQUIRED
  >
<!ELEMENT RequiredCompetence (Competence) >
  <!ATTLIST RequiredCompetence
    RequiredLevel CDATA #REQUIRED
  >
<!ELEMENT Competence EMPTY>
  <!ATTLIST Competence
    Identifier ID #REQUIRED
    Name CDATA #REQUIRED
    Description CDATA #IMPLIED
  >
<!ELEMENT Employee (PersonalData, Role,
EmployeeCompetence*)>
  <!ATTLIST Employee
    Identifier ID #REQUIRED
  >
<!ELEMENT EmployeeCompetence (Competence)>
  <!ATTLIST EmployeeCompetence
    CurrentLevel CDATA #REQUIRED
  >
<!ELEMENT PersonalData (firstName, lastName,
gender, birthDate, birthPlace, nationality+,
knownLanguage*, address+, phoneNumber+,
emailAddress+, note?)>
<!ELEMENT firstName (CDATA)>
<!ELEMENT LastName (CDATA)>
<!ELEMENT gender (Male|Female)>
<!ELEMENT birthDate (CDATA)>
<!ELEMENT birthPlace (CDATA)>
<!ELEMENT nationality (CDATA)>
<!ELEMENT knownLanguage (CDATA)>
<!ELEMENT address (CDATA)>
<!ELEMENT phoneNumber (CDATA)>
<!ELEMENT e-mailAddress (CDATA)>
<!ELEMENT note (CDATA)>
```

Fig. 3. The DTD describing the Ontology of a Skills Manager Agent.

- inserting, deleting and updating organization roles and competencies;
- inserting, deleting and updating employees' roles and competencies;
- querying the *skill map*, for example: searching for the most suitable employees to take a specific role or evaluating employee competence gaps computing, for each employee competence, the difference between the current competence level and the competence level required for holding a specific role.

Ontology. The ontology of a generic *SMA*, encoded according to the DTD⁶ shown in Fig. 3, consists of an XML document storing information about the *skill map SM* of the organization. The skill map *SM* con-

⁶A DTD is used here because it is more compact and readable than an XML Schema, however, an equivalent XML Schema was defined and used in the system implementation phase for defining and validating agent's ontologies.

```
<!ELEMENT CLOAssistantOntology (SkillMap,
LearningHistory*)>

<!-- SkillMap is defined in SMA Ontology -->
<!-- LearningHistory is defined in SAA
Ontology -->
```

Fig. 4. The DTD describing the Ontology of a CLO Assistant Agent.

sists of a set of roles, competencies and employees. A *role* is described by an identifier, its name and its required competencies. Each *required competence* specifies, for a particular role, the minimum level of knowledge required for a competence to fit that role. A *competence* is described by an identifier, its name and its description. Finally, an *employee* is characterized by an identifier, its personal data, its role and its competencies (with an associated *competence level*).

3.3.2. The CLO assistant agent

Provided Services. A generic *CLO* supports a Chief Learning Officer in managing the *skill map* of an organization and defining a learning strategy in terms of roles and required competencies. *CLO* provides the following services:

- managing roles and competencies: *CLO* supports the user in defining roles, associated competencies and required knowledge levels;
- managing employees' profile, role and competence levels;
- suggesting the most suitable employees to have a specific role assigned;
- defining priorities or/and temporal constraints associated with competencies that have to be acquired by employees;
- showing the learning activities history of each employee.

Ontology. The ontology of a generic *CLO* consists of an XML document storing information about the *skill map SM* of the organization and the *learning history lh_i* for each employee *Em_i* as shown in Fig. 4.

3.3.3. The Student Assistant Agent (SAA)

Provided Services. A generic Student Assistant Agent *SAA_i* is associated with a student (employee) *Em_i* and assists him in filling his competence gaps relevant to his roles. *SAA_i* provides *Em_i* with the following services:

- showing the gaps between his current competencies and the competencies required for his role in the organization, possibly associated with priorities or/and temporal constraints; this task is carried out in cooperation with the Skills Manager Agent;

- evaluating a specific competence gap by *pre-assessment* tests and finding or creating a courseware for filling that competence gap; this task is carried out in cooperation with the Learning Paths Agent;
- enriching and modifying a courseware according to feedbacks that users provide to optimize the acquisition of needed competencies; this task is carried out in cooperation with the Learning Paths Agent;
- managing information about current learning activities of the student.

Ontology. The ontology of a generic *SAA_i*, associated with a student *Em_i*, consists of an XML document storing information about the *user profile up_i* of *Em_i*. A *profile up_i* consists of a *learning history*, *login information* and a set of *session log*. A *learning history* consists of a set of *coursewares* and *assessments*. An *assessment* is a test conceived for evaluating whether a student has got a specific competence level, a *courseware* is a specific Learning Path that a user has to follow to acquire a competence and/or filling a competence gap. Each *courseware* consists of a set of selected Learning Objects. For each selected Learning Object, *SAA_i* stores information about its use, like download status, start learning timestamp and completion percentage. A *session log* consists of a set of *user actions*. The ontology of a *MASEL SAA* agent is described by the DTD shown in Fig. 5.

3.3.4. The Learning Paths Agent (LPA)

Provided Services. A Learning Paths Agent deals with the creation of learning paths suitable to fill the competence gaps of an employee *Em_i* supported by a Student Assistant Agent *SAA_i*. *LPA* provides *SAA_i* with the following services:

- providing a *pre-assessment* test allowing a specific competence gap to be evaluated;
- selecting and composing the Learning Objects suitable to build a learning path;
- enriching and modifying a learning path according to feedbacks that the user provides, for example, by answering to tests and assessments.

Moreover, the *LPA* might inform the *UPA* Agent about the lack of some Learning Objects needed to build specific learning paths.

Ontology. The ontology of a generic *LPA* consists of an XML document storing information about the *skill map SM* of the organization, the Learning Objects available for building learning paths, the *course-*

wares already built to allow users to acquire specific competencies, the *assessment* available for evaluating whether a student has a competence at a specific level and, for each employee Em_i , his *learning history* lh_i . The ontology of a *MASEL LPA* is described by the DTD shown in Fig. 6.

```
<!ELEMENT StudentAssistantOntology
(UserProfile)>
<!ELEMENT UserProfile (Employee,
LearningHistory, LoginInformation,
SessionLog*)>
<!ELEMENT LoginInformation EMPTY>
<!-- ATTLIST LoginInformation
Username CDATA #REQUIRED
Password CDATA #REQUIRED
-->
<!ELEMENT SessionLog (UserAction*)>
<!-- ATTLIST SessionLog
SessionId ID #REQUIRED
-->
<!ELEMENT UserAction EMPTY>
<!-- ATTLIST UserAction
ActionDescription CDATA #REQUIRED
-->
<!ELEMENT LearningHistory
(Courseware*, Assessment*)>
<!-- ATTLIST LearningHistory
Employee IDREF #REQUIRED
-->
<!ELEMENT Courseware (SelectedLO+)>
<!ELEMENT Assessment (assessmentItem+)>
<!-- ATTLIST Assessment
Competence IDREF #REQUIRED
EvaluatingLevel CDATA #REQUIRED
-->
<!ELEMENT SelectedLO (lom,
LearningObjectStatus?)>
<!ELEMENT LearningObjectStatus EMPTY>
<!-- ATTLIST LearningObjectStatus
Downloaded (true|false) #REQUIRED
StartLearningTimestamp CDATA #IMPLIED
CompletionPercentage CDATA #IMPLIED
-->
<!-- lom is defined in IMS Specification -->
<!-- assessmentItem is defined in IMS
Specification -->
```

Fig. 5. The DTD describing the Ontology of a Student Assistant Agent.

```
<!ELEMENT LearningPathOntology
(SkillMap, lom*, Courseware*,
Assessment*, LearningHistory*)
-->
<!-- lom is defined in IMS Specification -->
<!-- SkillMap is defined in SMA Ontology -->
<!-- LearningHistory is defined in SAA
Ontology -->
<!-- Courseware is defined in SAA Ontology
-->
<!-- Assessment is defined in SAA Ontology
-->
```

Fig. 6. The DTD describing the Ontology of a Learning Paths Agent.

3.3.5. The Content Agent (COA)

Provided Services. A Content Agent *COA* is specialized in managing a Learning Object Database. *COA* acts as a wrapper between the learning objects database and the rest of *MASEL* agents. *COA* agent assists a Learning Paths Agent *LPA* in building a courseware and a Chief Content Officer assistant agent *CCO* in managing learning contents providing the following services:

- inserting, deleting and updating Learning Objects, Assessments and associated descriptions;
- querying the Learning Object Database by accessing Learning Object Metadata (*LOM*).

Ontology. The ontology of a *COA* consists of an XML document storing information about the Learning Objects currently stored in the Learning Object Database and the *assessment* available for evaluating if a student has a competence at a specific level. Each *assessment* and each Learning Object is described by an XML document validated with respect to an XML Schema established by the IMS standard [20]. The ontology of a *MASEL COA* is described by the DTD shown in Fig. 7.

3.3.6. CCO Assistant Agent

Provided Services. A generic *CCO* supports a Chief Content Officer in managing learning contents that will be used to build learning paths by the *LPA* agent. It provides the following services:

- inserting, deleting and updating Learning Objects, Assessments and associated descriptions; this task is carried out in cooperation with the Content Agent (that acts as an interface between the *CCO* agent and the *LOs* database).

Ontology. The ontology of a generic *CCO* consists of an XML document storing information about the *skill map SM* of the organization, the *assessments* available for evaluating a specific competence level and the Learning Objects available for building learning paths. The ontology of a *MASEL CCO* is described by the DTD shown in Fig. 8.

```
<!ELEMENT ContentAgentOntology (Assessment*,
lom*)>
-->
<!-- Assessment is defined in SAA Ontology
-->
<!-- lom is defined in IMS Specification -->
```

Fig. 7. The DTD describing the Ontology of a Content Agent.

```
<!ELEMENT CCOAssistantOntology (SkillMap,
lom*,Assessment*)>

<!-- SkillMap is defined in SMA Ontology -->
<!-- lom is defined in IMS Specification -->
<!-- Assessment is defined in SAA Ontology
-->
```

Fig. 8. The DTD describing the Ontology of a CCO Assistant Agent.

3.3.7. The Project Manager assistant Agent (PMA)

Provided Services. A generic *PMA* supports a Project Manager in selecting the most suitable employees for carrying out a given project. A *PMA* provides the following services:

- specifying a project in terms of required roles and competencies;
- suggesting the most suitable employees for carrying out a specific project.

These services are carried out in cooperation with the Skills Manager Agent.

Ontology. The ontology of a generic *PMA* consists of an XML document storing information about the *projects* and *skill map SM* of the organization. A *project* is associated with a *project manager*, a *team*, and the *project constraints*. A *project manager* is the *employee* who leads the project. A *team* consists of a set of *employees*; each *employee* holds a role in the project (*project role*) and carries out a given *workload*. A *project role* requires a set of *competencies* to be held. The *project constraints* consists of a *budget*, a *duration* (in days), a *start date*, and a *workload*. The ontology of a *MASEL PMA* is described by the DTD shown in Fig. 9.

3.3.8. The User Profile Agent (UPA)

Provided Services. A User Profile Agent stores useful information about all the registered users. For instance: personal data, log-in data, current and past learning activities. A *UPA* provides the following services:

- managing user log-in to the system;
- managing user profile information;
- updating the competence levels of an employee according to his learning activities; this task is carried out in cooperation with the Student Assistant Agent, associated with the employee, and with the Skills Manager Agent;
- informing the CCO about the lack of learning objects needed to build learning paths for filling specific competence gaps.

```
<!ELEMENT ProjectManagerOntology (Project*,
SkillMap)>
<!ELEMENT Project (ProjectManager, Team,
ProjectConstraints)>
<!ELEMENT ProjectManager EMPTY>
  <!ATTLIST ProjectManager
    Employee IDREF #REQUIRED
  >
  <!ELEMENT Team ((ProjectRole,Employee,WorkLoad)+)>
  <!ELEMENT ProjectRole (RequiredCompetence+)>
    <!ATTLIST ProjectRole
      Identifier ID #REQUIRED
      Name CDATA #REQUIRED
    >
  <!ELEMENT ProjectConstraints (Budget,
Duration, StartDate, WorkLoad)>
  <!ELEMENT WorkLoad (CDATA)>
  <!ELEMENT Budget (CDATA)>
  <!ELEMENT Duration (CDATA)>
  <!ELEMENT StartDate (CDATA)>

<!-- SkillMap is defined in SMA Ontology -->
<!-- Employee is defined in SMA Ontology -->
<!-- RequiredCompetence is defined in SMA
Ontology -->
```

Fig. 9. The DTD describing the Ontology of a Project Manager assistant Agent.

Ontology. The ontology of a generic *UPA* consists of an XML document storing the *profiles* of the registered users. Moreover, it contains a set of CCO Information. This information is transmitted by the *LPA* agent to the *CCO* agent through the *UPA* agent in order to notify that the LOs database is not currently including some needed LOs dealing with a required subject. The *UPA* agent is involved in this interaction because *CCO* is not supposedly always logged in the system.⁷ In such a way, when the *CCO* assistant agent logs in, *UPA* informs it about this absence, in order to provide for the enrichment of the LOs database. The ontology of a *MASEL UPA* agent is described by the DTD shown in Fig. 10.

3.3.9. Interaction protocols

The basic interactions among the agents in *MASEL* can be derived by the following use case:

- a Chief Learning Officer requires his *CLO* agent to suggest the most suitable employees for taking a specific role and/or assigns a specific role to an employee, in this case the *SMA* agent evaluates the employee competence gaps according to the assigned role;
- when the employee Em_i logs in, his Student Assistant Agent SAA_i shows the gaps between his current competencies and the competencies required for his role in the organization; these are specified along with priorities or/and temporal

⁷ Assistants agents log in depending on user behaviour.

```

<!ELEMENT UserProfileOntology (UserProfile*,
CCOInformation*)>
<!ELEMENT CCOInformation EMPTY>
<ATTLIST CCOInformation
  Competence IDREF #REQUIRED
  WarningNote CDATA #REQUIRED
>

<!-- UserProfile is defined in SAA Ontology
-->

```

Fig. 10. The DTD describing the Ontology of a User Profile Agent.

constraints related to the competencies that need to be acquired. This task is carried out in cooperation with the SMA agent. When Em_i decides to fill a specific competence gap k , say $Cgap_{i_k}$, SAA_i asks the LPA agent to perform three tasks:

- providing a *pre-assessment* test by which to evaluate $Cgap_{i_k}$ more precisely;
- selecting and composing Learning Objects $\{LO_{k_1}, \dots, LO_{k_n}\}$ in order to build a suitable learning path Lp_k ;
- enriching and modifying Lp_k according to feedbacks that the user provides.

These tasks are carried out by the LPA agent in cooperation with the Content Agent (COA) that provides the LPA with the suitable Learning Objects;

- given a learning path to follow, Em_i works, interacting with his SAA_i agent, both on-line or off-line. To support this feature both the SAA_i agent and UPA agent store all the information about the current learning activities of the employee. Storing a copy of the user learning activities in the UPA agent knowledge base has two further important advantages: (i) a complete user profile is always available in the system and can be used by CLO for analysis on employee activities (ii) increases system recoverability.⁸ Periodically, the SAA_i agent and UPA agent synchronise their knowledge about Em_i ;
- UPA , in cooperation with SMA , updates the current competence level Col_{i_k} of Em_i according to the learning activities that have been carried out;
- eventually, if a Project Manager needs to build a team for carrying out a specific project, he interacts with his PMA agent. The PMA agent, collaborating with the SMA agent, suggests a set of

employees possessing the competencies needed to carry out the required tasks.

All these interactions conform to the standard FIPA request protocol [15] but one. In fact, in accordance with [16], a Student Assistant Agent exploits a special defined learning path building protocol to request a Learning Paths Agent to build a learning path (Fig. 11). The corresponding interactions can be described as follows: SAA_i asks the LPA agent to build a learning path Lp_k suitable to fill a specific competence gap (*buildCourse* request message), say $Cgap_k$. To this end, first LPA agent asks the SMA agent about the current competence level of Em_i (*getEmployeeCompetence* request message). After that, the LPA agent receives this information from the SMA agent, it asks the COA to build, by querying the LOM database, a set of “potentially useful LOs” to consider for the construction of Lp_k (*searchLO* request message). Note that, if the COA cannot find an appropriate set of LOs (*failure* message), the process terminates and the LPA agent informs the UPA about the lack of appropriate Learning Objects.⁹ Otherwise, after that the LPA agent receives this set from the COA , it interacts with the SAA_i agent to complete building Lp_k .

In particular, from this set, the LPA agent chooses a *first set* Ls_{k1} of LOs (*sendSet* inform-done message) to send to the SAA_i . The user, interacting with the SAA_i agent, chooses, from Ls_{k1} , an LO and SAA_i communicates the user choice to the LPA agent (*selectioInfo* inform message).

Therefore, the LPA agent is able to build a new set Ls_{k2} of LOs to propose to the user for continuing the construction of the learning path according to user preferences (*sendSet* inform message). The process described above is repeated until a complete course is built¹⁰ and the user accepts it (*courseSelected* inform message).

It is worth noting that this specification supports different strategies for the construction of learning paths [16].

3.4. System implementation

A prototype tool was developed partially implementing *MASEL* using JADE (Java Agent Development Framework) [21], a software framework allowing

⁸ SAA_i knowledge base can be recovered by also exploiting information stored in the UPA knowledge base and vice versa.

⁹ When the CCO Assistant Agent logs in, the UPA informs it about this lack asking for the Learning Object database to be enriched.

¹⁰ In which case, the LPA agent sends an empty set Ls_{kj} to SAA_i agent.

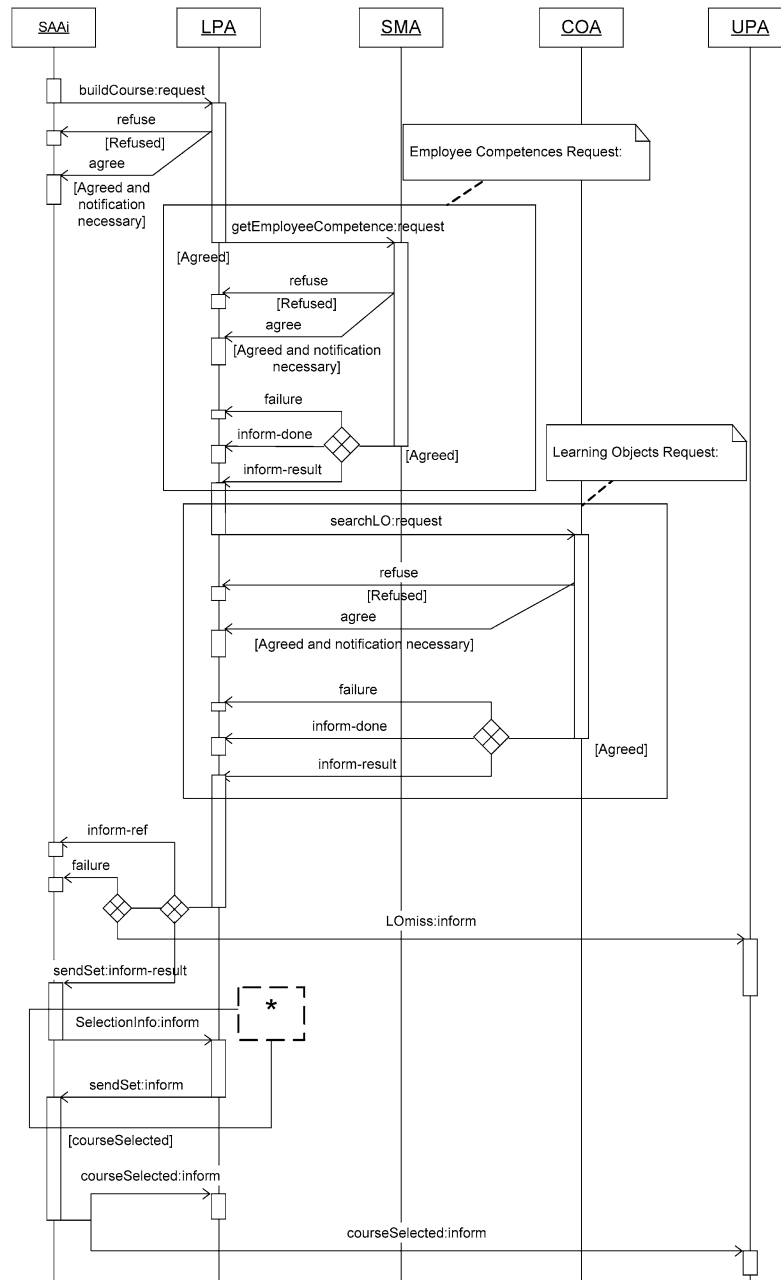


Fig. 11. A diagram showing the learning paths building protocol.

for the development of multi-agent systems and applications conforming to FIPA standards [15].

In particular the following features were implemented:

- a Web based user interface for *SAA* agents.
- *UPA* agent for users login/logout.
- *SMA* agent to manage user competencies.

- *COA* and *LPA* agents to implement the learning path building process [16].

The *SAA* user interface has been developed by using Java Servlet and Java Server Pages (JSP).¹¹ To specify the reasoning capability of the *LPA* and *COA* agents disjunctive Datalog under the answer set seman-

¹¹They run inside Jakarta Tomkat 4.1 [42] a Servlet/JSP container.

tic [18] has been exploited. That specification was implemented by using the DLV system [10,13]. The interested reader can refer to [16] for information about the underlying logic model and the employed logic programming techniques. The system is available at <http://si.deis.unical.it/~garro/masel>.

3.4.1. A logic-based implementation of the Learning Path Building Process

In this section the learning path building process implemented in MASEL will be described using an example.

First of all, a logic-based specification of the agents' knowledge is given, then a simplified version of the logic programs exploited to implement the *COA* and *LPA* reasoning capabilities are described.¹²

There are three basic entities which are involved in the learning path building process: subjects (like, e.g., "Functions in C++"), learning objects (like, e.g., "A tutorial on Objects and Classes in C++"), and competencies (like, e.g., "C++ Programmer"). They are represented in *MASEL*, together with the relationships holding among them, by a set of ground (variable-free) facts for the following set of logic predicates:

- subjects are stored in the unary predicate *subject*;
- learning objects are described by ternary predicate *lObject*; *lObject(o123, "A tutorial on Objects and Classes in C++", 2)* says that the LO with identifier o123 has title "A tutorial on Objects and Classes in C++" and a student takes 2 hours to work with it (this is an estimated time which is called *duration* of the learning object);
- competencies (that can be *RequiredCompetencies* or *EmployeeCompetencies*, see Fig. 3) are stored in the binary predicate *Skill*. Skills have different levels of competence (e.g., beginner, intermediate, expert); *skill("C++ Programmer", expert)* says that exists a skill named "C++ Programmer" at expert level;
- users' knowledge is stored in the binary predicate *knows*; *knows(u100, "Functions in C++")* specifies that the student u100 knows the subject "Functions in C++";
- prerequisites are stored in predicate *requires*; *requires("Inheritance in C++", "Object and Classes in C++")* specifies that the subject "Object and Classes in C++" is a prerequisite for the subject "Inheritance in C++";

- subjects to be learned to get a given skill are specified by the predicate *partOf*; *partOf("C++ Programmer", beginner, "Functions in C++")* specifies that a user has to study (also) subject "Functions in C++", in order to get the skill "C++ Programmer" at beginner level;
- the subjects treated by learning objects are specified by predicate *deals*; *deals(o123, "Objects and Classes in C++")* specifies that the LO identified by o123 deals with the subject "Object and Classes in C++"¹³; after studying o123, the student will get the knowledge of subject "Object and Classes in C++".

In order to illustrate how those notions can be exploited to build learning paths, a simple knowledge base compliant with the described logic model is reported in Fig. 12. The execution of a logic program which is able to collect the LOs which are potentially *useful* to get the desired skill constitutes the most important reasoning task performed by the *COA* agent.

First of all the set of subjects the user has to know in order to get the desired skill (called *needed subjects*) are determined by using the following two logic rules:

```
neededSubject(SubjA) ←
    input(User, Skill, Lev, TMax),
    partOf(Skill, Lev, SubjA),
    not knows(User, SubjA).

neededSubject(SubjA) ←
    requires(SubjB, SubjA),
    neededSubject(SubjB),
    input(User, _, _, _),
    not knows(User, SubjA).
```

Here, the "input" predicate encodes the user and the skill (including the skill level) for which an appropriate building path should be built. This predicate is not stored permanently in the knowledge base but, each time a new user request is made, it is inserted on the spot before starting the path building process.

The intuitive reading of the above rules is the following: a given subject *SubjA* is a *needed subject* if the user does not know it, and one of the following conditions hold: (i) *SubjA* is part of the definition of the required skill level *Skill*; (ii) *SubjA* is a prerequisite for a needed subject *SubjB*.

¹²The reader is assumed to be familiar with Disjunctive Logic Programming syntax and semantics [13], and is referred to [18,10] for an introduction on these topics.

¹³In *MASEL*, a fine granularity for learning objects was considered, assuming that learning objects are *atomic*. In other words, LO are designed to deal only with one subject.

```

% Subjects

subject("Functions in C++").
subject("Objects and Classes in C++").
subject("Inheritance in C++").
subject("UML Modeling").

% Learning Objects

lObject(o1, "C++ Functions tutorial", 10). %
a tutorial
lObject(o2, "C++ Functions", 20). % a book
section
lObject(o3, "Preliminary on C++ Objects", 5).
% a tutorial
lObject(o4, "Objects and Classes in C++",
10). % a book section
lObject(o5, "Inheritance in C++, basic
concepts", 15). % a tutorial
lObject(o6, "Inheritance in C++, advanced
features", 20). % a book section
lObject(o7, "UML Modeling for C++
programmers", 10). % a book section
lObject(o8, "UML Advanced Modeling and C++",
11). % a book section

% Deals relation

deals(o1, "Functions in C++").
deals(o2, "Functions in C++").
deals(o3, "Objects and Classes in C++").
deals(o4, "Objects and Classes in C++").
deals(o5, "Inheritance in C++").
deals(o6, "Inheritance in C++").
deals(o7, "UML Modeling").
deals(o8, "UML Modeling").

% Skills

skill("C++ Programmer", beginner).
skill("C++ Programmer", intermediate).
skill("C++ Programmer", expert).

% Skills definitions

partOf("C++ Programmer", beginner, "Functions
in C++").
partOf("C++ Programmer", intermediate,
"Object and Classes in C++").
partOf("C++ Programmer", intermediate,
"Inheritance in C++").
partOf("C++ Programmer", expert, "UML
Modeling").

% Requires relation

requires("Inheritance in C++", "Objects and
classes in C++").
requires("Objects and Classes in C++",
"Functions in C++").

% The u1 user's knowledge.

knows(u1, "Functions in C++").

```

Fig. 12. An instance of the knowledge base.

As an example, suppose that the user *u1* would like to reach the *intermediate* skill level for the competence “C++ Programmer” spending maximum 25 hours. The fact *input(u1, “C++ Programmer”, intermediate, 25)* is added to the knowledge base of Fig. 12, and the

above logic rules derive that the subjects the user needs to study are: “Object and Classes in C++” and “Inheritance in C++”.

The learning objects dealing with some needed subject are collected in the predicate *usefulLO*:

```

usefulLO(ID, Title, Duration) ←
    neededSubject( Subj ),
    deals( LO, Subj ),
    lObject( ID, Title, Duration ).

```

The selected objects are, in general, only a subset of the LOs that will be eventually used to build the desired learning path by the *LPA* agent (in which only one LO per subject will be considered).

Continuing with the example, the learning objects selected for user *u1* are: *o3, o4, o5, o6*.

It is worthwhile noting that only the useful LOs selected in this phase will be considered in the next phase of the process which is carried out by the *LPA*. The following disjunctive logic program is the heart of the *LPA* reasoning capability.

Following the well-known *Guess and Check* [10] programming methodology, the learning paths are built specifying the search space by using a disjunctive rule (Guess), and singling out the admissible ones by using some integrity constraint (Check). A learning path is said to be *admissible* if it allows the user to get the specified skill within the specified time-bound.

```

inPath(ID) ∨ notInPath(ID)
    ← usefulLO( ID, _, _ ).

← neededSubject( Sub ), not dealt( Sub ).
← inPath( X ), inPath( Y ), deals( X, S1 ),
    deals( Y, S2 ), S1 <> S2.
← #sum{ T : inPath( X ), lObject( X, _, T ) }
    <= B, input( _, _, _, B ).

dealt( Sb ) ← input( User, _, _, _ ), knows
    ( User, Sb ).
dealt( Sb ) ← inPath( LO ), deals( LO, Sb ).

```

The disjunctive rule intuitively means that an LO selected in the first phase is in the path or not (guess stage). The first constraint ensures that all the needed subjects are dealt with in the path. The second constraint ensures that the path is not redundant (it does not contain two learning objects dealing with the same subject) and the third one ensures that the path does not exceed the specified time bound.

The solutions to the above program correspond one-to-one to the admissible paths of the specified input. Sets of LOs can be “sequentialized” to a learning path because the *requires* relation induces a natural order between subjects and corresponding learning objects. The system exploits this relation to suggest to the user the learning objects that can be “safely” included in the path.

Eventually, in our example, the admissible computed learning paths are: {o3, o5}, {o4, o5}, {o3, o6}.

3.4.2. The graphical user interface

In this section the user interface of the implemented MASEL prototype will be described through an example.

A generic Student (or Employee) interacts with the system by using its Student Assistant Agent (*SAA*). Suppose that he is connected to the system to fill his competence gaps. First of all the user asks the system for the list of his skill gaps (Fig. 13). At this point the user asks the system to build a specific learning path by clicking on the link “build course” which corresponds to the skill gap he wants to fill. By using the form shown in Fig. 14, the user can tune the parameters of the learning path building process specifying the desired course duration (maximum time) t_{max} , and the skill level he wants to reach. When the user clicks the “submit button”, the *SAA* agent asks the *LPA* agent to build a Learning path Lp able to meet the selected learning objective, and which complies with the specified parameters. The learning path process starts following the protocol described in Section 3.3.9.

After the *LPA* Agent has received the list of “potentially useful Learning Objects (LOs)” from the *COA* agent (as described in Section 3.4.1), it can start the second phase of the learning path building process. During this phase, the *LPA* agent cooperates with the *SAA* agent helping the user to build the personalized learning-path Lp . During this interaction the user is driven by the system to choose the learning objects in Lp as follows:

- the *SAA* agent shows the user the set of learning objects suggested by the *LPA* Agent¹⁴ (see Fig. 15);
- the user selects a particular learning object from the proposed set and confirms the choice (by clicking the “submit button”).

¹⁴This set is computed by the *LPA* agent by using the DLV system accordingly with [16].

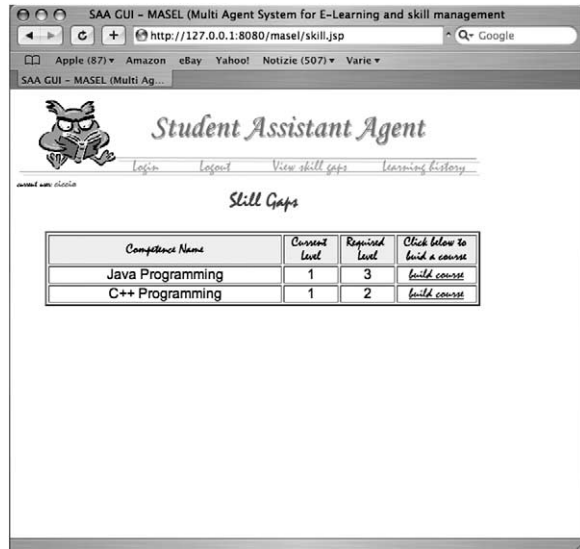


Fig. 13. A user is choosing the skill gap to be filled.

The process stops when a complete learning-path is built and the *SAA* GUI shows the “Course Built” message.

Moreover, the *SAA* GUI permits to the user to modify his choices clicking the “change selection” button. Note that, in accordance with [16], at each interaction, the user can choose only admissible LOs. In other words, the system guarantees that the choices allow the specified skill gap to be filled, and they comply with the parameters specified by the user (e.g., the imposed time bound).

3.5. System testing and evaluation

During the development of the prototype, a number of tests were performed in order to verify the behavior of the system.

The testing tools shipped with JADE [21] were intensively used. This suite contains a number of specialized “testing” agents which were conceived in order to check different behavioral aspects of a multi-agent system.

In particular, the agent interactions were tested by using the *Sniffer* agent, and resulted compliant with the protocols described in Section 3.3.9; the internal status of each agent was checked in a number of selected interaction by using the *Introspector* agent. Moreover, a test suite implemented by using JUnit [25], conceived in order to test the java code implementing each agent behavior has been successfully passed without problems.

Competence name	C++ Programming
Current level	1
Required Level	2
Goal level:	2
Course duration (in hours)	1

Fig. 14. A user is choosing the learning path building parameters.

Title	A course on C++ Programming
Goal	Meeting level 2
Max time (in min)	20
LO N°1	
LO N°2	Object and Classes on C++
LO N°3	Introduction to C++

Fig. 15. A user is building a learning path.

In order to evaluate the system in a real-world scenario, the prototype was then experimentally exploited by *Exeura s.r.l.* [12], a spin-off company providing Knowledge Management solutions. In particular, *MASEL* was exploited for the training of a project team, composed of five employees, involved in the *G4B-Government for Business* project, a project concerning the development of a service-based e-government system. The Learning Object database was filled with about two hundred Learning Objects available in the company. The prototype system performed

reasonably well (some minor bugs were notified, and the learning path building required a reasonable time). The employees declared that the system helped them to fill their competence gaps. The company is now evaluating the possibility of launching an e-learning product based on *MASEL*.

This experience demonstrated the applicability of *MASEL* in realistic scenarios.

4. Related work

In [38] the authors propose *IDEAL* (Intelligent Distributed Environment for Active Learning), a multi-agent system for active distance learning. *IDEAL* consists of: (i) a personal agent, handling the profile of a learner (i.e., the background knowledge, the interests and the learning style); (ii) a course agent, managing both the materials and the teaching technique of a course; (iii) a teaching agent, behaving as an intelligent tutor for a learner. In *IDEAL*, course materials are decomposed into small components called *Lecturelets*. These are XML documents containing JAVA code; they are dynamically assembled to cover course topics according to learner progress.

IDEAL and *MASEL* share various similarities; indeed, e.g., both of them are XML based and exploit user modeling techniques. The main differences between them are the following: (i) the Curriculum Sequencing Activity of *IDEAL* and the Learning Path construction of *MASEL* are based on different philosophies and strategies; (ii) *IDEAL* exploits non-standard and complex constructs for managing course contents (i.e., *Lecturelets*) whereas *MASEL* uses the concept of learning object, following the *IMS* standard; (iii) *IDEAL* does not explicitly manage competence skill gaps.

In [50] an approach for exploiting Web-mining techniques to build a software agent supporting e-learning activities is presented. The proposed agent acts as a recommender system, i.e., it can produce both suggestions (helping the learner to better navigate through on-line materials) and shortcuts (helping the learner to quickly find needed resources). In order to perform all these activities, the system intensively exploits a user profile taking into account the learner access history.

MASEL and the system proposed in [50] both exploit a user profile and operate by constructing the most appropriate learning path. The main differences existing between *MASEL* and the system proposed in [50] are the following: (i) [50] presents a single-agent architec-

ture whereas *MASEL* is multi-agent; (ii) the learning path construction is based on data mining techniques in [50], whereas it is automated-reasoning based (by exploiting disjunctive logic programming system DLV) in *MASEL*.

[3] proposes an e-learning platform, named *KnowledgeSea*. The core of the system proposed in [3] is a self-organized hyperspace map, i.e., an automatically-built map that provides a concise navigation support for a relatively large learning hyperspace. The map may help a user to find and access on-line educational resources.

The main differences of *KnowledgeSea e-learning platform* and *MASEL* are the following: (i) the self-organized hyperspace map provides a more flexible mechanism for selecting learning objects than that of *MASEL*; however, it does not handle pre-requisite relationships possibly holding among learning objects; (ii) *KnowledgeSea* does not handle the construction of a complete learning path able to fill a specific competence gap; vice versa, in *MASEL*, the *LPA* agent has been conceived exactly to achieve this purpose; (iii) *KnowledgeSea* also take into account the device a user is exploiting for accessing educational resources, which is not considered in *MASEL*.

In [39] the authors propose a handheld learning device and an appropriate software infrastructure to support child education. The main components of the proposed architecture are: (i) a learning manager, which stores a local cache of learning objects extracted by a repository and exploits specific software agents to search and organize learning objects, (ii) a communication manager, which creates direct voice and data communication channels for disseminating learning materials and handles resource sharing. [39] develops a technology for assisting individuals and groups to learn anytime and anywhere.

Similarly to *MASEL*, in [39] learning materials follow the IMS standard and might have different multimedia formats, but *MASEL* does not approach the problem of adapting the learning objects distribution to the device characteristics; on the other hand *MASEL* is specifically conceived to deal with enterprise e-learning and offers skills management services; these aspects are not dealt with in [39].

In [36] a multi-agent prototype called CITS (Confidence Intelligent Tutoring Agent) is proposed. CITS approach aims at being adaptive (i.e., it can adjust learning materials to meet user needs) and dynamic (i.e., it adapts the offered service to user current behaviour). CITS architecture consists of five kinds of

agents, namely: (i) a Cognitive Agent, that creates a model for each learner, representing his level and learning style; (ii) a Behaviour Agent, that monitors the learner's behaviour during their interaction with the system for improving the model produced by the Cognitive Agent; (iii) a Guide Agent, that selects and classifies information potentially useful for the learner; (iv) an Information Agent, that searches over the Internet for extra information required by the learner and, (v) a Confidence Agent, that is in charge of strengthening the confidence between the learner and the system. In CITS, learning information is fragmented in simple pieces called *knowledge targets*; these might have different multimedia formats.

CITS and *MASEL* are both XML-based multi-agent systems supporting the dissemination of learning materials and offering some "freedom degrees" in the learning paths definition; however CITS knowledge targets and *MASEL* learning objects are different in their characteristics and purposes. *MASEL* learning objects metadata are IMS compliant, stored in a database handled by *COA* agent. *MASEL* learning objects are selected and classified by *CCO* agent according to the organization learning plan. CITS knowledge targets are selected and also acquired browsing the Web according to single user needs.

In [40] the system ELETROTUTOR is proposed. ELETROTUTOR consists of the following agents: (i) a Pedagogical Agent, performing learning activities, such as the distribution and the dissemination of examples and exercises; (ii) a Remote Agent, managing the communication between the learner and the system; (iii) a Communication Agent, handling agent communications, and (iv) a Student Model Agent, handling a student profile and exploiting it for performing the learning activities.

Both *MASEL* and ELETROTUTOR are multi-agent systems and both of them adapt the dissemination of learning contents to user profiles. However, ELETROTUTOR does not handle the construction of complete learning paths and does not explicitly manage competence skill gaps.

In [8] an overview of the *ELENA* system is presented. In *ELENA* there are heterogeneous learning resource providers possibly characterized by different goals and knowledge backgrounds. These providers can autonomously edit learning resources that are, then, stored in their own learning repositories. The information stored in each learning repository is represented by means of OWL [45]; the exploitation of this formalism allows learning resources in different learn-

ing repositories to be integrated into a common knowledge base called *open learning repository*. In *ELENA* each user is associated with a profile. When he is interested in a learning resource, he submits a query and *ELENA* properly manipulates this query by introducing additional constraints based on user preferences (*query rewriting service*). After this, it processes the modified query and returns links to suitable learning resources satisfying user requirements (*link generation service*).

ELENA and *MASEL* share some similarities; specifically, both of them manage detailed user profiles and operate in a distributed e-learning scenario. *ELENA* represents and manages both learning resources and user profiles by means of Semantic Web technologies; *MASEL*, instead, uses the IMS specification for describing learning objects and XML for representing user profiles.

Summarizing, it is worth pointing out that the main difference between *MASEL* and the various systems cited above is that *MASEL* is specifically conceived to deal with enterprise e-learning (is a learning platform able to help enriching, sharing and circulating organization knowledge). *MASEL* supports learning requirements of organization working in a project-shaped perspective as an important component of the enterprise knowledge management system. Given a project specification *MASEL* would be able to suggest a project team, to evaluate human resources competence gaps and to reduce them by creating personalized learning paths.

5. Conclusions

To evaluate the contribution that agent technologies can offer in the e-learning and skills management context, a Multi Agent System for E-Learning and Skill Management (*MASEL*) was designed and prototyped.

MASEL is strongly based on a continuous interaction among involved agents; this activity is facilitated by the choice of XML both to represent agent ontologies and to handle data exchange. In particular, the learning paths building problem was approached defining a general interaction protocol (see Section 3.3.9) which allows agents to adopt different path building strategies.

A prototype tool was developed implementing *MASEL* using *JADE* [21].

The reasoning capabilities of *MASEL* agents involved in the learning path building process is implemented using the DLV system [14]. The learning

paths adopted building strategy complies with the logic model defined in [16].

The system was experimented within a R&D project put forward by *Exeura s.r.l.*, a spin-off Knowledge Management Company.

As far as future work is concerned, it is planned to complete *MASEL* implementation and to continue to extensively evaluate the system on real application cases not only in *Exeura s.r.l.* but also in other companies. In addition, it is planned to improve cooperation among the agents of the system by extending it to support other tasks like *optimal team building* and the definition of *career path* for employees taking into account gaps between their possible future roles and current competencies.

References

- [1] B. Bauer, J.P. Muller and J. Odell, Agent UML: a formalism for specifying multiagent interaction, in: *Agent-Oriented Software Engineering*, Springer, Berlin, 2001, pp. 91–103.
- [2] R. Ben-Eliyahu and R. Dechter, Propositional semantics for disjunctive logic programs, *Annals of Mathematics and Artificial Intelligence* **12**(1–2) (1994), 53–87.
- [3] P. Brusilovsky and R. Rizzo, Map-based horizontal navigation in educational hypertext, in: *Proceedings of 13th ACM Conference on Hypertext and Hypermedia (Hypertext 2002)*, College Park, MD, 2002, ACM, pp. 1–10.
- [4] F. Buccafurri, N. Leone and P. Rullo, Enhancing disjunctive datalog by constraints, *IEEE Transactions on Knowledge and Data Engineering* **12**(5) (2000), 845–860.
- [5] M. Cossentino, P. Burrafato, S. Lombardo and L. Sabatucci, Introducing pattern reuse in the design of multi-agent systems, in: *Agent Technologies, Infrastructures, Tools, and Applications for E-Services, NODE 2002 Agent-Related Workshop*, H. Tianfield, R. Kowalczyk, J.P. Müller and R. Unland, eds, Erfurt, Germany, 2002, Revised Papers, LNAI 2592, Springer, 2003, pp. 107–120.
- [6] DARPA Agent Markup Language + Ontology Inference Layer, [<http://www.daml.org/language/>].
- [7] J. Dix, G. Gottlob and V.W. Marek, Reducing disjunctive to non-disjunctive semantics by shift-operations, *Fundamenta Informaticae* **28**(1–2) (1996), 87–100.
- [8] P. Dolog, N. Henze, W. Nejdl and M. Sintek, Personalization in distributed e-learning environments, in: *Proceedings of the International World Wide Web Conference – Alternate track papers & posters (WWW Alt.Š04)*, New York, NY, USA, 2004, pp. 170–179.
- [9] G. Duncan, R. Beckett and F. Soliman, A comprehensive organizational model for Enterprise Knowledge Management, in: *Proceedings of the 1st International Conference on Enterprise Information Systems*, Setubal, Portugal, 1999, pp. 783.
- [10] T. Eiter, W. Faber, N. Leone and G. Pfeifer, Declarative problem-solving using the DLV system, in: *Logic-Based Artificial Intelligence*, J. Minker, ed., Kluwer Academic Publishers, 2000, pp. 79–103.

- [11] T. Eiter, G. Gottlob and H. Mannila, Disjunctive Datalog, *ACM Transactions on Database Systems* **22**(3) (1997), 364–418.
- [12] Exeura – Knowledge Management Solutions, [http://www.exeura.it/].
- [13] W. Faber, N. Leone and G. Pfeifer, Experimenting with heuristics for answer set programming, in: *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI) 2001*, Morgan Kaufmann Publishers, Seattle, WA, USA, pp. 635–640.
- [14] W. Faber and G. Pfeifer, since 1996, DLV homepage, [http://www.dlvsystem.com/].
- [15] FIPA, 2003. Foundation for Intelligent Physical Agents Specifications, [http://www.fipa.org/].
- [16] A. Garro, N. Leone and F. Ricca, Logic based agents for e-learning, in: *Proceedings of the IJCAI Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, Acapulco, Mexico, 2003, pp. 36–45.
- [17] A. Garro and L. Palopoli, An XML multi-agent system for e-learning and skill management, in: *Agent Technologies, Infrastructures, Tools, and Applications for E-Services, NODE 2002 Agent-Related Workshop*, H. Tianfield R. Kowalczyk, J.P. Muller and R. Unland, eds, Erfurt, Germany, 2002, Revised Papers, LNAI 2592, Springer, 2003, pp. 283–294.
- [18] M. Gelfond and V. Lifschitz, Classical negation in logic programs and disjunctive databases, *New Generation Computing* **9**(3–4) (1991), 365–386.
- [19] IEEE-ITSC, 2003. Lom working draft, [http://ltsc.ieee.org/].
- [20] IMS, 2003. Instructional management system global learning consortium specifications, [http://www.imsproject.org/].
- [21] JADE, 2003. Java agent development framework project, [http://jade.cse.it/].
- [22] N.R. Jennings, An agent-based approach for building complex software systems, *Communications of the ACM* **44**(4) (2001), 35–41.
- [23] N.R. Jennings, K.P. Sycara and M.J. Wooldridge, IA roadmap of agent research and development, *Journal of Autonomous Agents and Multi-Agent Systems* **1**(1) (1998), 7–36.
- [24] N.R. Jennings and M.J. Wooldridge, Agent technology: foundations, applications, and markets, *Journal of Artificial Societies and Social Simulation* **2**(4) (1999).
- [25] JUnit, Testing resources for extreme programming, [http://www.junit.org/].
- [26] P. Kaipa, Design organizations that learn: An executive guide to learning, in: *Chinmaya Management Review*, [http://www.mithya.com/learning/designorg.htm].
- [27] R. Klimecki and L. Hermann, What causes organizations to learn? in: *Proceedings of the 3rd International Conference on Organizational Learning*, Vol. 2, Lancaster, England, pp. 551–577.
- [28] N. Leone, P. Rullo and F. Scarcello, Disjunctive stable models: Unfounded sets, fixpoint semantics and computation, *Information and Computation* **135**(2) (1997), 69–112.
- [29] J. Lind, Issues in agent-oriented software engineering, in: *Proceedings of the First International Workshop on Agent-Oriented Software Engineering (AOSE)*, LNCS 1957, Springer, Berlin, 2001, pp. 45–58.
- [30] J. Lobo, J. Minker and A. Rajasekar, *Foundations of Disjunctive Logic Programming*, The MIT Press, Cambridge, Massachusetts, 1992.
- [31] Y. Malhotra, Organizational learning and learning organizations: An overview, 1996, [http://www.brint.com/papers/orglrng.htm].
- [32] I. Nonaka and H. Takeuchi, *The Knowledge Creating Company*, Oxford University Press, 1995.
- [33] D.E. O’Leary, Enterprise Knowledge Management, *IEEE Computer Journal* **31**(3) (1997), 54–61.
- [34] Ontology eXchange Language, [http://www.ai.sri.com/~pkarp/xol/].
- [35] Ontology markup language, [http://www.ontologos.org/OML/OML%200.3.htm].
- [36] M.A. Razek, C. Frasson and M. Kaltenbach, A confidence agent: Toward more effective intelligent distance learning environments, in: *Proceedings of the International Conference on Machine Learning and Applications (ICMLA ’02)*, Las Vegas, Nevada, USA, 2002, CSREA Press, pp. 187–193.
- [37] Resource Description Framework, [http://www.w3.org/TR/rdf-schema/].
- [38] Y. Shang, H. Shi and S. Chen, An intelligent distributed environment for active learning, in: *Proc. of the International Conference on World Wide Web (WWW 2001)*, Hong Kong, 2001, ACM Press, pp. 308–315.
- [39] M. Sharples, D. Corlett and O. Westmancott, The design and implementation of a mobile learning resource, *Personal and Ubiquitous Computing* **6**(3) (2002), 220–234.
- [40] R.A. Silveira and R.M. Vicari, Improving interactivity in e-learning systems with multi-agent architecture, in: *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002)*, Malaga, Spain, LNCS, Springer, 2002, pp. 466–471.
- [41] Simple HTML Ontology Extension, [http://www.cs.umd.edu/projects/plus/SHOE/].
- [42] Tomcat Web site, The Apache Jakarta Project, [http://jakarta.apache.org/tomcat/].
- [43] Trinity College, E-Learning Sub Group Report, [http://www.tcd.ie/CLT/elearning/definition.htm].
- [44] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, Third Edition, Addison-Wesley Professional, Boston, MA, 2003.
- [45] Web Ontology Language, [http://www.w3.org/TR/owl-features/].
- [46] D.A. Wiley, Learning object design and sequencing theory, [http://davidwiley.com/papers/dissertation/].
- [47] M. Wooldridge and P. Ciancarini, Agent-oriented software engineering: the state of the art, in: *Proceedings of the Agent-Oriented Software Engineering, First International Workshop (AOSE 2000)*, Revised Papers, LNCS 1957, Springer, 2001, pp. 1–28.
- [48] M. Wooldridge and N.R. Jennings, Intelligent agents: Theory and practice, *The Knowledge Engineering Review* **10**(2) (1995), 115–152.

- [49] M. Wooldridge, N.R. Jennings and D. Kinny, The Gaia methodology for agent-oriented analysis and design, *Journal of Autonomous Agents and Multi-Agent Systems* **3**(3) (2000), 285–312.
- [50] O.R. Zaiane, Building a recommender agent for e-learning systems, in: *Proceedings of the International Conference on Computers in Education (ICCE 2002)*, Auckland, New Zealand, IEEE Press, 2002, pp. 55–59.

Copyright of AI Communications is the property of IOS Press and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.