A Knowledge Production Protocol for Cooperative Development of Learning Objects

Juan M. Dodero
DEI Laboratory
Universidad Carlos III de Madrid
Avda. de la Universidad, 30
28911 Leganés
Madrid, Spain
dodero@inf.uc3m.es

Miguel A. Sicilia
DEI Laboratory
Universidad Carlos III de Madrid
Avda. de la Universidad, 30
28911 Leganés
Madrid, Spain
masicilia@inf.uc3m.es

Elena García-Barriocanal
Computer Science Department
Universidad de Alcalá de Henares
28871 Alcalá de Henares,
Madrid, Spain
elena.garciab@uah.es

ABSTRACT

In a distributed *eLearning* environment, the development of learning objects becomes a cooperative task. We consider learning objects as knowledge pieces, which are subject to the management processes of acquisition, delivery, and production. We present a two-tier architecture for cooperative knowledge production tasks. In our model, knowledge-producing agents are arranged into knowledge domains or *marts*, and a distributed negotiation protocol is used to consolidate knowledge produced in a mart. The architecture can be extended to a multiple-tier architecture, where consolidated knowledge can be negotiated in higher-level foreign marts. This approach is applied to cooperative development of learning objects, although it is proposed to be also applicable to other knowledge production tasks.

Categories and Subject Descriptors

H.5.3 [**Information Systems**]: Group and Organization Interfaces—*Computer-supported cooperative work, Theory and models*; K.3.1 [**Computing Milieux**]: Computer Uses in Education—*Collaborative learning*

General Terms

Algorithms

Keywords

Knowledge production, Learning objects, Collaborative development, Multi-agent architectures

1. INTRODUCTION

A learning object can be defined as a set of learning contents, integrated with course structure and sequencing (following LTSA terminology [2]). In a more generic sense, a learning object is any entity, digital or non-digital, which can be used, re-used or referenced during technology-supported learning. The production of learning objects becomes a common task in the industry of Internet-based learning services —what is often referred to as *e-Learning*—. In this work, we are considering learning objects as knowledge pieces, which are subject to the management processes of acquisition, delivery, and production [16].

1.1 A Case Study of Learning Objects Development

The *IMS (*Instructional Management Systems*) Global Learning Consortium* has recently released the public draft version 1.1 of the IMS Content Packaging Specification [7], that provides the functionality to describe and package learning materials, such as an individual course or a collection of courses, into interoperable, distributable packages

Some commercial toolkits, like Microsoft LRN Toolkit[1], Peer3[2] and ToolBook II[3], that provide implementations of the IMS Content Packaging Specification, have been released. Collaborative development of learning objects is not supported in such tools, which present a unipersonal vision of the creation, edition, viewing, and testing of learning objects.

When constructing a learning object, the builder can use a tool like *Microsoft LRN*, as shown in figure 1. If the learning object is being jointly developed, another user may wish to contribute, making some modification in the structure of the course, or adding some object to the course contents. The IMS specification defines the structure of the learning object by *manifest* XML files, as depicted in figure 2. Proposed additions or modifications are transformed into changes to the manifest file.

While developing a learning object, two or more authors can cooperate with the exchange of proposals, which are implemented as changes to the manifest file. Before any change is considered as consolidated, it must be negotiated between the participating authors. We will restrict our example to

[1] `http://www.microsoft.com/elearn/support.asp`
[2] `http://www.peer3.com/text/software/software.html`
[3] `http://home.click2learn.com/products`

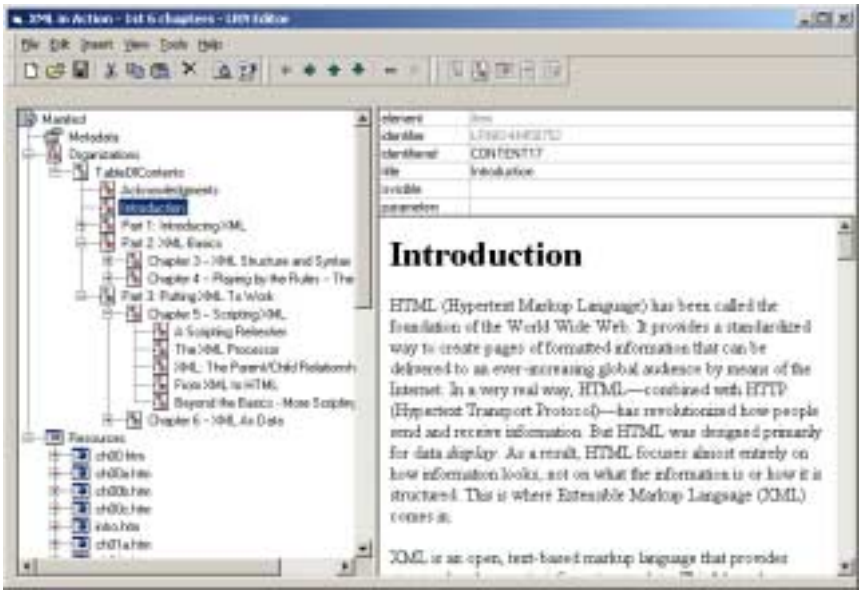

Figure 1: Microsoft LRN Learning Object Editor, ©Microsoft Corp.

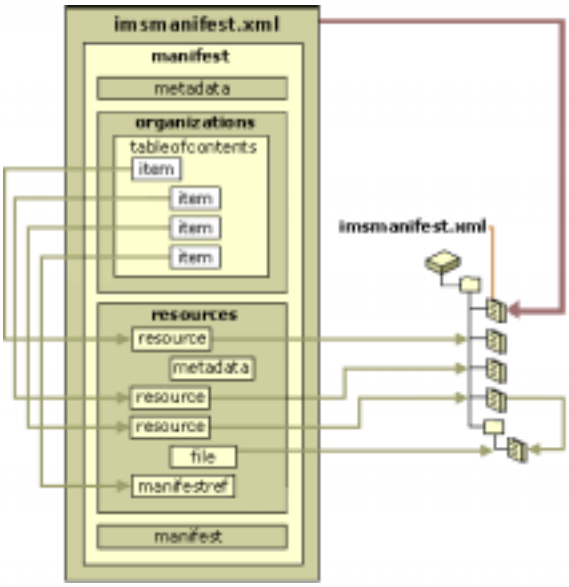

Figure 2: IMS manifest file structure, ©Microsoft Corp.

the `tableofcontents` structure of the `organizations` section in the manifest file. Nevertheless, collaborative development can be also extended to `resources` or `metadata` sections.

Let us consider the development of a learning object named "Introduction to XML" with Microsoft LRN Toolkit. In a given moment, an author knows the currently consolidated knowledge (see figure 3) and elaborates a new proposal (see figure 4). There may be several negotiation processes initiated, each one affecting a section of the learning object, that can be negotiated separately (v.g., `tableofcontents`, `resources`, and `metadata`). Then, the author submits a proposal, including the differences between both files, and refering to `tableofcontents` negotiation. The rest of collaborating authors receive and evaluate the proposal, according to a set of previously agreed criteria, like those described in section 3.3. Then, a negotiation protocol is executed by every author until the proposal is eventually accepted, or substituted by a further elaborated proposal. This process continues until an agreement is reached or some degree of consensus is achieved.

1.2 Cooperative development of learning objects

Since learning objects can be considered as knowledge, their cooperative development is a process that can be knowlegdemanaged. Knowledge management can be considered as a discipline that affects every knowledge process carried out in a given organization (i.e., a group of people who work together). According to Swanstrom [16], knowledge processes can be broadly classified into three categories, that are enumerated according to the common timing of implementation.

Acquisition Knowledge is a subset of information that has been extracted, filtered and formatted in a specific way. Once the usefulness of information is proved, information becomes knowledge. The goal of knowledge acquisition is to extract knowledge from available information sources.

Delivery and learning Knowledge needs to be constantly updated and delivered to the right places at the right time. Learning is a continuous process that can be oriented by the delivery of knowledge interesting to a person or community of users.

Production Knowledge becomes a network of ideas, plans, or artifacts, that are produced by both experts and

```
<?xml version="1.0" ?>
- <manifest identifier="MANIFEST1" xmlns="x-schema:IMS_CONTENTv1p0.xdr">
  + <metadata>
  - <organizations>
    - <tableofcontents identifier="TOC1">
        <item identifier="LRNID44450656" identifierref="CONTENT16" title="Acknowledgments" />
        <item identifier="LRNID44450752" identifierref="CONTENT17" title="Introduction" />
      + <item identifier="LRNID44450848" title="Part 1: Introducing XML">
      - <item identifier="LRNID44451872" title="Part 2: XML Basics">
        + <item identifier="LRNID44451936" identifierref="CONTENT37" title="Chapter 3 -- XML Structure
            and Syntax">
        + <item identifier="LRNID44452320" identifierref="CONTENT43" title="Chapter 4 -- Playing by the
            Rules -- The DTD">
        </item>
      - <item identifier="LRNID44452608" title="Part 3: Putting XML To Work">
        + <item identifier="LRNID44452672" identifierref="CONTENT51" title="Chapter 5 -- Scripting XML">
        + <item identifier="LRNID44453248" identifierref="CONTENT59" title="Chapter 6 -- XML As Data">
        </item>
      </tableofcontents>
    </organizations>
  + <resources>
  </manifest>
```

Figure 3: Initial state of a manifest file while developing a learning object

```
<?xml version="1.0" ?>
- <manifest identifier="MANIFEST1" xmlns="x-schema:IMS_CONTENTv1p0.xdr">
  + <metadata>
  - <organizations>
    - <tableofcontents identifier="TOC1">
        <item identifier="LRNID44450656" identifierref="CONTENT16" title="Acknowledgments" />
        <item identifier="LRNID44450752" identifierref="CONTENT17" title="Introduction" />
      - <item identifier="LRNID44450848" title="Part 1: Introducing XML">
        + <item identifier="LRNID44450912" identifierref="CONTENT20" title="Chapter 1 -- Understanding
            Markup Languages">
        + <item identifier="LRNID44451392" identifierref="CONTENT27" title="Chapter 2 -- Enter XML">
        </item>
      - <item identifier="LRNID44451872" title="Part 2: XML Basics">
        + <item identifier="LRNID44451936" identifierref="CONTENT37" title="Chapter 3 -- XML Structure
            and Syntax">
        + <item identifier="LRNID44452320" identifierref="CONTENT43" title="Chapter 4 -- Playing by the
            Rules -- The DTD">
        </item>
      - <item identifier="LRNID44452608" title="Part 3: Putting XML To Work">
        + <item identifier="LRNID44452672" identifierref="CONTENT51" title="Chapter 5 -- Client-side
            Scripting XML">
          <item identifier="MANIFEST1_ITEM1" title="Chapter 6 -- Server-side scripting XML" isvisible="1"
            parameters="" />
        + <item identifier="LRNID44453248" identifierref="CONTENT59" title="Chapter 7 -- XML As Data">
        </item>
      </tableofcontents>
    </organizations>
  + <resources>
  </manifest>
```

Figure 4: Proposed state of a manifest file while developing a learning object

users. A cooperation model is advisable to coordinate such knowledge production tasks, and to ensure that knowledge production effort is not duplicated.

In a highly distributed environment —for instance, the faculty staff in a virtual university—, cooperative development of learning objects becomes harder, since the holding of synchronous —physical or virtual— meetings is quite less frequent. The exchange of ideas between members of the distributed workgroup is an asynchronous process, where participants may keep their own pace within the interchange. Although, a group of agents can jointly, asynchronously develop learning objects if they coordinate their creation activities.

An structured model of cooperation between knowledge-producing agents is presented in section 2. In our model, agents are arranged into cooperative knowledge marts, that are described in section 2. Knowledge production in a mart is achieved by argumentation-based negotiation (section 3). Section 4 presents the distributed negotiation protocol to coordinate knowledge-producing agents, and outlines some implementation issues. Finally, conclusions and future work are described in section 5.

2. COOPERATIVE KNOWLEDGE PRODUCTION ARCHITECTURE

Knowledge-producing agents can operate into disjoint domains or knowledge marts that we call *zocos*[4], as shown in figure 5. An agent can operate in a foreign market by using an intermediate proxy agent. Proxy agents act as mart representatives in foreign marts, according to the *proxy* design pattern [5], so that interaction between marts is not tightly coupled. To facilitate cooperation between domains, marts can be structured in a hierarchical way, as depicted in figure 6.

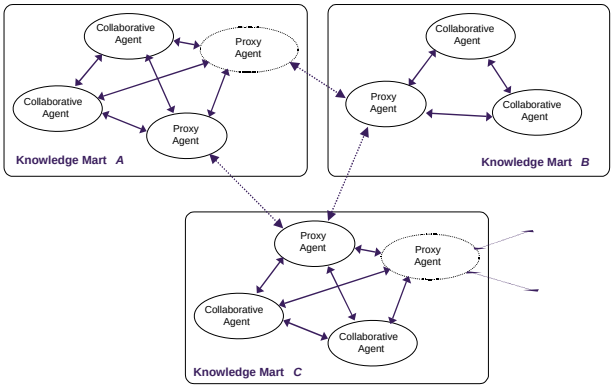

Figure 5: Cooperative knowledge marts

In this architecture, cooperation domains can be modeled as *knowledge marts*, and marts can be arranged into *knowledge warehouses*.

Cooperative knowledge mart A Cooperative Knowledge Mart (CKM) is a distributed group of cooperative agents that try to produce a piece of knowledge in a given domain. When knowledge produced in a mart can affect agents' performance in some other domain, a special proxy agent can act as representative in a foreign mart.

[4] *Zoco* is the spanish word for *souk*, an open-air market in an Arabian city

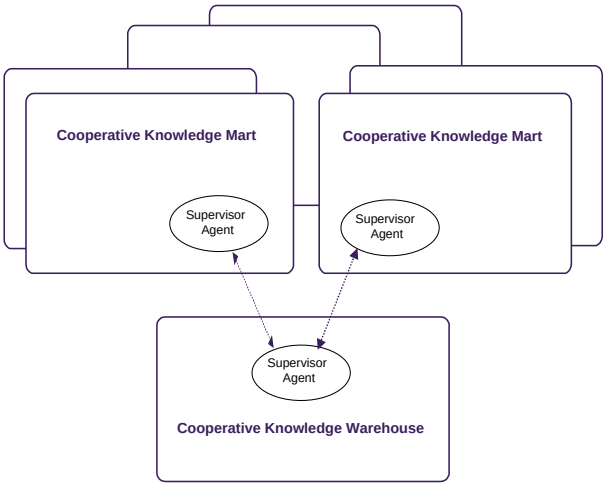

Figure 6: Cooperative knowledge warehouse

Cooperative knowledge warehouse A Cooperative Knowledge Warehouse (CKW) is the place where knowledge produced in foreign marts is merged in a structured fashion. Two or more CKMs can negotiate using representatives in a common CKW.

Because of simplicity, we define a two-tier cooperative work architecture. Nevertheless, it can be easily extended to a multiple-tier architecture, where cooperative agents collaborate in lower-level domains to consolidate some knowledge, before they try to collaborate or compete in higher-level domains.

Collaborative development of a number of learning objects that make up an in-development curriculum is not a trivial task. Changes or additions in some learning material can affect several courses that depend on it. Sometimes, two or more teaching staff members have to negotiate the inclusion of some content in a given course under his/her responsibility. In this case, the architecture of several marts, depicted in figure 6 can be helpful to structure the negotiation.

3. AGENT NEGOTIATION IN A KNOWLEDGE MART

Negotiation is an essential kind of interaction in multi-agent knowledge systems. Agents do not usually enjoy an inherent control over each other. Thus, the usual way to influence one another is persuasion. In some cases, a *persuadee* agent needs a few arguments to behave according to the persuader directives. In other cases, the persuadee is hardly determined to accept persuader's proposals. Then, the persuadee has to be convinced to change its beliefs, goals, or preferences, in order to accept —perhaps modified— proposals.

The minimum requirement to negotiate is that agents can build and deliver proposals, which can be accepted or rejected. An example is the Contract Net Protocol (CNP) [15]. The collaboration protocol is more sophisticated when recipients have a chance to build counterproposals that alter certain issues that were not satisfactory in the original proposal [13]. A more elaborated form of negotiation allows parties to send justifications or arguments along with proposals. Such arguments indicate why proposals should be accepted [17][11][12].

This section introduces an approach to design argumentative multi-agent architectures, where agents try to convince each other to accept a given knowledge in some domain. Agents are structured into knowledge marts, thus building up knowledge domains. The aim is to allow agents to *consolidate* knowledge continuously produced in a CKM. Knowledge consolidation in a CKM is the establishment of knowledge as accepted by every agent operating in the CKM. Agents can reach a consensus on the CKM-wide accepted knowledge by the exchange of messages.

3.1 Negotiation Principles

Negotiation between agents is carried out by exchanging proposals in a common language, like ACL [10]. Proposal interchange is directed by goals and necessities of the participating agents. Although the formalisation of agents' communication language and goals are not included as objectives of our model, this account is subject to a minimal set of conventions about the language and negotiation protocol:

1. Agent rationality is modeled in terms of *preference relationships* or *relevance functions* [4], in order to allow agents to evaluate and compare proposals.
2. Relevant aspects of negotiation can be modeled as *issues* and *values* that can be changed as negotiation progresses.
3. Agents can deliberate and achieve an internal *state*, thus recording the history of negotiation and their decisions.

An agent can be involved in several negotiation processes. The negotiation protocol described in section 4 keeps a separate negotiation where agents can participate.

3.2 Messages

The basic types of messages abstractions that can be exchanged between agents belonging to the same CKM are the following:

proposal(*k*,*n*) Given a negotiation process n, agents send a *proposal* message when they have a piece of knowledge k and wish it to be consolidated in the CKM.

consolidate(*k*,*n*) Agents send a *consolidate* message when they reach a given state in the negotiation protocol, for a previously submitted own proposal k to be accepted in a negotiation process n.

Sources and recipients of messages are not explicited as parameters in above messages, since they are concern of an underlying transport protocol that guarantees a reliable delivery. As well, message delivery to every agent in the same CKM has to be supported by some multicasting facility in the underlying transport.

3.3 Proposal Relevance

As stated above, agents rationality needs to be modeled in terms of preference relationships or relevance functions, in order to allow agents to evaluate and compare proposals. Linguistic-expressed preferences [6] can also be integrated in negotiation processes, as proposed in [1].

Next, we give some definitions used in the protocol described in section 4.

Proposal attributes Proposal attributes are elementary criteria to be considered when comparing proposals in a CKM. Some examples of proposal attributes are:

- Submitter's hierarchical level, useful when agents present different decision privileges in the CKM about the acceptance of proposals (v.g., lecturer vs. assistant in a faculty staff).
- Degree of fulfilment of a set of goals. For instance, before the development of a learning content, a set of educational objectives should be defined. In the case of corporate learning, these goals can be conducted by the training needs of the organization.
- Timestamp of the moment when a proposal was firstly submitted in the CKM (normally considered in the last case, when no other attribute can decide).

Proposal relevance The relevance of a proposal is defined as the set of proposal attributes considered at the moment of negotiation.

Proposal relevance function The relevance function $u(k)$ of a proposal k in a CKM returns a numerical value, dependent on attributes of k, in such a way that if $k_i \neq k_j$, then $u(k_i) \neq u(k_j)$.

Proposal preference relationship A proposal k_1 is preferred to another k_2 in a CKM, denoted as $k_1 \succ k_2$, if $u(k_1) > u(k_2)$.

4. NEGOTIATION PROTOCOL IN A CKM

Let $\mathcal{A}_\mathcal{M} = \{A_1, \ldots, A_n\}$ be a discrete set of cooperative agents, participating in a knowledge mart $\mathcal{M}$.

Rule 1 When A_i wants a knowledge piece k to be consolidated in $\mathcal{M}$, it sends a $proposal(k_i, n)$ to every agent in $\mathcal{M}$, initiating a new negotiation n. Then, A_i sets a timeout t_0 before confirming its proposal. During t_0, messages can arrive from any other agent A_j, with $(j \neq i)$, consisting of new proposals —maybe the original, though modified— referring to the same negotiation n.

Rule 2 If A_i does not receive any message referred to n during t_0, it considers that there is no agent against its proposal and it tries to ratify its proposal, by sending a $consolidate(k_i, n)$ to every agent in $\mathcal{M}$.

Rule 3 If A_i receives a $proposal(k_j, n)$ message from other agent A_j, referring to the same negotiation n, A_i evaluates the new proposal k_j. If $k_i \prec k_j$, then A_i sets a new time out t_1, waiting for proposal k_j to be ratified. Then, A_i proceeds as follows:

1. If A_i does not receive any proposal referred to negotiation n before t_1 expires, then A_i initiates back the protocol in Rule 1 with the same proposal k_i.
2. If A_i receives a $consolidate(k_j, n)$, with $k_j \succ k_i$, for $j \neq i$, before t_1 expires, and referring to the same negotiation n, then A_i gives up the initial proposal and the protocol finishes unsuccessfully.
3. If A_i receives a new $proposal(k_j, n)$, with $k_i \prec k_j$, it extends the timeout t_1.

We are considering two different timeouts, one for each phase that can be noticed in the negotiation process. T_0 is used for the distribution phase, that occurs after an agent submits a proposal by executing Rule 1. T_1 is used for the consolidation phase, that occurs if an agent that is in its distribution phase (t_0-waiting) receives a proposal that is evaluated as preferred.

In any moment, the reception of a message from another agent may provoke a momentary retraction from a previously submitted proposal, until a counter-proposal is elaborated. An agent that has not reached this state will be waiting for t_0 timeout. Then, if the agent receives a proposal that is evaluated as preferred, a new timeout t_1 is set to give it a chance. But if the preferred proposal is not eventually ratified, then the agent goes on about its aims and will try again to consolidate its own proposal.

A cooperative agent A_i can participate in several negotiation processes. Each negotiation process is handled separately, by initiating a new execution thread of the protocol.

4.1 Activation Events

In any moment, an agent A_i can be involved in a negotiation due to the arrival of a message from A_j referring to the same negotiation. The following rules describe the actions to undertake by agent A_j when it receives a message from another agent A_j.

- If A_i receives a $proposal(k_j, n)$ from A_j:

  Rule A If A_i sent a $proposal(k_i, n)$ and is waiting for t_0 timeout, then it can perform one of the following actions:
  - If $k_j \succ k_i$, then A_i acts in the same manner as in step 3 and waits for proposal k_j to be ratified or an alternative proposal to come.
  - If $k_j \prec k_i$, then A_i sends last proposal it sent in the negotiation process back to A_j, and extends t_0 timeout.

  Rule B If A_i sent a $proposal(k_i, n)$ and is waiting for t_1 timeout, then it can perform one of the following actions:
  - If $k_j \succ k_i$, then A_i acts in the same manner as in step 3-3 and waits for proposal k_j to be ratified.
  - If $k_j \prec k_i$, then A_i sends last proposal it sent in the negotiation process back to A_j, and extends t_1 timeout.

  Rule C If A_i did not sent any message referred to the same negotiation n, it does nothing.

- If A_i receives a $consolidate(k_j, n)$ from A_j:

  Rule A If A_i sent a $proposal(k_i, n)$ and is waiting for t_0 timeout, then it can perform one of the following actions:
  - If $k_j \succ k_i$, then A_i acts in the same manner as in step 3-2 and the protocol finishes unsuccessfully.
  - If $k_j \prec k_i$, then A_i sends last proposal it sent in the negotiation process back to A_j, and extends t_0 timeout.

  Rule B If A_i sent a $proposal(k_i, n)$ and is waiting for t_1 timeout, then it can perform one of the following actions:
  - If $k_j \succ k_i$, then A_i acts in the same manner as in step 3-2 and the protocol finishes unsuccessfully.
  - If $k_j \prec k_i$, then A_i acts in the same manner as in step 3-1 and initiates back the protocol with the same proposal.

  Rule C If A_i did not sent any message referred to the same negotiation n, it does nothing.

4.2 Protocol Variants

The negotiation protocol described above uses only two message types (i.e., proposal and consolidate). Some variants using additional message types can also be formulated:

retract(k, n) Agents can retract from a previous proposal k by issuing this message referred to a negotiation n.

substitute(k_1, k_2, n) Agents can replace a previously issued proposal k_1 by a new proposal k_2. It is equivalent to $retract(k_1, n)$ followed by $proposal(k_2, n)$.

reject(k, n) Agents can express with this message their refusal for a proposal k without necessity to formulate and issue a new proposal.

4.3 Implementation

We have built a prototype implementation of the protocol, that uses message multicasting facilities provided by Java *Aglets* [9] development framework. In the end-user version of the protocol, messages will be delivered by e-mail, with parameters coded into the SMTP message header. A concurrent versioning system will track the changes and commit the consolidated proposals into a common repository.

5. CONCLUSIONS AND FUTURE WORK

This work presents a model to develop a collaborative multi-agent architecture, applied to collaborative development of learning objects. There are currently some popular protocols of cooperation knowledge used heavily by multi-agent systems. These protocols can be classified into the following approaches:

- Top-down methodologies try to design domain specific agent systems, either from a market-based approach (v.g., the contract net protocol or CNP [14]), or from an organizational approach (v.g., the facilitator protocol or FP [3]). With top-down methodologies, it becomes difficult to separate cooperation level knowledge from problem-solving domain level knowledge. Jennings [8] claimed the existence of a social level knowledge that provides an abstract framework for comparing and analyzing all types of multi-agent systems.
- Bottom-up methodologies aim to generate families of components that can be assembled to build collaborative agent systems in a more reusable fashion. When the members of a multi-agent system are scattered over a very large scope on the Internet, Yuan and Wu [18] sector them into few territories and duplicate the social level knowledge in each territory. This approach could lead to the problem of inconsistency.

The architecture presented here is a bottom-up approach to the design of cooperative multi-agent systems. Every CKM holds responsibilities on some domain level knowledge, while cooperation level knowledge interfaces to other domains are well-defined. The structuring of knowledge marts can help to reduce inconsistencies between agent territories. The cooperative approach presented in this work is also applicable to other knowledge production tasks, as software development, specially in analysis and design phases. Nevertheless, further validation is needed to assess the usefulness of the protocol in different scenarios, and we are conducting tests on the impact of the number of agents in the overall effectiveness of the model. Our work does not consider yet how knowledge marts are set up. As a future work, knowledge mart generation and the participation of agents in CKMs are proposed to be dynamic, depending on agents' ontology-based expressed interests.

6. REFERENCES

[1] M. Delgado, F. Herrera, E. Herrera-Viedma, and L. Martínez. Combining numerical and linguistic information in group decision making. *Information Sciences*, 7:177–194, 1998.

[2] F. Farance and J. Tonkel. Draft Standard for Learning Technology – Learning Technology Systems Architecture (LTSA). Technical report, Learning Technology Standards Committee of the IEEE Computer Society, Nov. 2000.

[3] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Proceedings of the 3rd Int. Conf. on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, Maryland, 1994. ACM Press.

[4] P. C. Fishburn. *Utility Theory for Decision Making*. Robert E. Krieger Publishing Co., Huntington, New York, 1969.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1994. ISBN 0-201-63361-2.

[6] F. Herrera, E. Herrera-Viedma, and J. Verdegay. A linguistic decision process in group decision making. *Group Decision and Negotiation*, 5:165–176, 1996.

[7] IMS Global Learning Consortium. IMS Content Packaging Specification. Technical report, IMS Global Learning Consortium, Inc., Dec. 2000.

[8] N. R. Jennings and J. R. Campos. Towards a social level characterisation of socially responsible agents. *IEEE Proceedings on Software Engineering*, 144(1):11–25, 1997.

[9] D. B. Lange and M. Oshima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1998.

[10] J. Mayfield, Y. Labrou, and T. Finin. Desiderata for agent communication languages. In *AAAI Spring Symposium on Information Gathering*, 1995.

[11] S. D. Parsons and N. R. Jennings. Negotiation through argumentation – A preliminary report. In *Proc. Second Int. Conf. on Multi-Agent Systems*, pages 267–274, Kyoto, Japan, 1996.

[12] H. Sawamura and S. Maeda. An argumentation-based model of multi-agent systems. In H. Jaakkola and H. Kangassalo, editors, *Proceedings of the 10th European-Japanese Conference on Information Modelling and Knowledge Bases*, number A28, pages 96–109, Saariselkä, Finland, May 08–11 2000.

[13] C. Sierra, P. Faratin, and N. Jennings. A service-oriented negotiation model between autonomous agents. In *Proc. 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, pages 17–35, Ronneby, Sweden, 1997.

[14] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. In *IEEE Transactions on Computers*, number 12 in C-29, pages 1104–1113, 1980.

[15] R. G. Smith and R. Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):61–70, Jan. 1981.

[16] E. Swanstrom. *Knowledge Management: Modeling and Managing the Knowledge Process*. John Wiley & Sons, 1999.

[17] K. Sycara. Persuasive argumentation in negotiation. *Theory and Decision*, 28:203–242, 1990.

[18] S. T. Yuan and Z. L. Wu. An infrastructure for engineering cooperative agents. *International Journal of Software Engineering*, 10(6):681–711, 2000.