# A Multi-Agent Architecture and Protocol for Knowledge Production: A Case Study for Participative Development of Learning Objects

*Juan Manuel Dodero* *Ignacio Aedo* *Paloma Díaz-Pérez*

*Laboratorio DEI, Universidad Carlos III de Madrid, Spain*

dodero@inf.uc3m.es aedo@ia.uc3m.es pdp@inf.uc3m.es

## Abstract

In a distributed *eLearning* environment, the development of learning objects is a participative task. We consider learning objects as knowledge pieces, which are subject to the management processes of acquisition, delivery, creation and production. A multiple-tier architecture for participative knowledge production tasks is introduced, where knowledge-producing agents are arranged into knowledge domains or *marts*, and a distributed interaction protocol is used to consolidate knowledge that is produced in a mart. Knowledge consolidated in a given mart can be in turn negotiated in higher-level foreign marts. The proposed architecture and protocol are applied to coordinate the development of learning objects by a distributed group of authors.

**Keywords:** multi-agent systems, knowledge management, learning objects.

## Introduction

According to the Learning Technology Standards Committee (LTSC) terminology (Farance & Tonkel, 2001), a learning object is any entity, digital or non-digital, which can be used, re-used or referenced during technology-supported learning. In a more generic sense, a learning object can be defined as any digital resource that can be reused to support learning (Wiley, 2002).

The production of learning objects becomes a participative task in the industry of Internet-based learning services —what is often referred to as *e-Learning*—. The term *learning* is used in this paper and related works according to the definition given by the Association for Learning Technology (ALT, 1996), who understands learning technology in a broad conceptual sense as the systematic application of a body of knowledge to the design, implementation and evaluation of learning resources. Our interpretation is not referred to cybernetic definitions of learning (Bateson, 1972; Von Goldammer & Kaehr, 1988) and machine learning.

### *A Knowledge Management Vision*

We consider a learning object as a set of learning contents, integrated with a given course structure and sequencing. From a component-oriented approach, learning objects are curriculum units that can be assembled to build higher-level learning contents. These units are pieces of knowledge, and we can cope with them by means of knowledge management techniques.

Knowledge management is a discipline that affects every process carried out in a given organization or a group of people who work together. According to Swanstrom (1999), knowledge management processes can be broadly classified into three categories, which 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 acquisition is to extract knowledge from available information sources.

### Delivery

Knowledge needs to be constantly updated and delivered to the right places at the right time. This is a continuous process that is devoted to delivering knowledge that is 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 users. It is advisable to coordinate knowledge production tasks, and to ensure that such a production effort is not duplicated.

Since learning objects can be knowledge-managed, their development is subject to the processes of acquisition, delivery, and production (Swanstrom, 1999). The main issue of this work has to do with *participative production* or *creation* of knowledge.

## *Participative Knowledge Production*

When a group of people is participatively creating (or producing) a complex object, it is advisable to establish a set of rules to coordinate its development. This is the situation, for instance, when several people are building some educational material (for instance, courseware or some other learning resource).

We will follow a concrete example for a better explanation of the problem. Let's suppose two instructors who are designing respective modules, which will be part of the same learning object. During the design stage, the necessity to develop a sub-element —i.e, some figure or diagram— with similar purpose can be detected by both instructors. Each one usually has his/her own pace of work in developing the common element. Also, they can be differently skilled in that work. If the development process is not appropriately coordinated, the following problems can arise:

- An instructor could get her work crushed, depending on the required speed and quality of the design, in comparison to her partner's competency.
- When speed is more important than quality, a more elaborated and reusable product can be readily thrown away.
- In the best case, effort will be duplicated in several phases of the project.

Therefore, the coordination of participative production of knowledge meets the following objectives:

- Bring together participants' different pace of creation.
- Take advantage of participants' different skills in the problem domain and the tools that are managed.

- Reduce the number of conflicts provoked by interdependencies between in-production knowledge components.
- In a more general sense, avoid duplication of effort.

## *An Agent-oriented Approach*

Agents have been proven as a helpful tool for the coordination of human people who are performing a given task (Maes, 1994). Agent interaction protocols govern the exchange of a series of messages among agents, i.e. a conversation. There are currently some popular interaction protocols 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 —e.g, the contract net protocol (Smith, 1980)— or from an organizational approach —e.g, the facilitator protocol (Finin et al., 1994).— With top-down methodologies, it becomes difficult to separate cooperation-level knowledge from problem-solving domain level knowledge. Jennings and Campos (1997) claim for 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 (2000) sector them into few territories and duplicate the social-level knowledge in each territory.

Depending on the protocol, a given relationship among agents is set up:

- When the agents commit themselves to common goals, their interaction is *coordinated*.
- In cases where agents have conflicting goals or are simply self-interested, the objective of the protocol is to maximize the payoffs of the agents, and their interaction is *negotiated*.
- In any case, the relationship established among agents can be *competitive* or *cooperative*.
- If there is a dominance relation among groups of agents, the relationship can be *participative* —i.e, subordinates can participate in decision making.

The architecture presented in this work is a bottom-up, multi-agent approach, and our working hypothesis is that a group of agents can help in the participative production of knowledge, by coordinating their creation activities. Therefore, different agents can act as representatives of knowledge-producing actors, according to the following principles:

- Agents can be structured into separable knowledge domains of interaction. This structuring reflects the knowledge differences between developers.
- A dynamic re-thinking of the structure of interactions in different domains can help to reduce the inter-dependencies during the process.

Once introduced the problems of participative knowledge production and our working thesis, we summarize the rest of this paper. A structured model of interaction between knowledge-producing agents is presented. In our model, agents are arranged into participative knowledge domains that we call *marts*. Knowledge production in a mart is participative, and a distributed interaction protocol to coordinate knowledge-producing agents is presented. A case study of participative development of learning objects is depicted. Finally, some conclusions and future work are described.

# A Participative Knowledge Production Architecture

In our architecture, knowledge-producing agents can operate within the boundaries of a specific domain or knowledge mart, as shown in figure 1. Nevertheless, interaction among different domains is also supported through a number of proxy agents. In order to facilitate interaction between domains, marts can be structured in a hierarchical way. In this architecture, domains can be modeled as *knowledge marts*, and marts are arranged into *knowledge warehouses*.

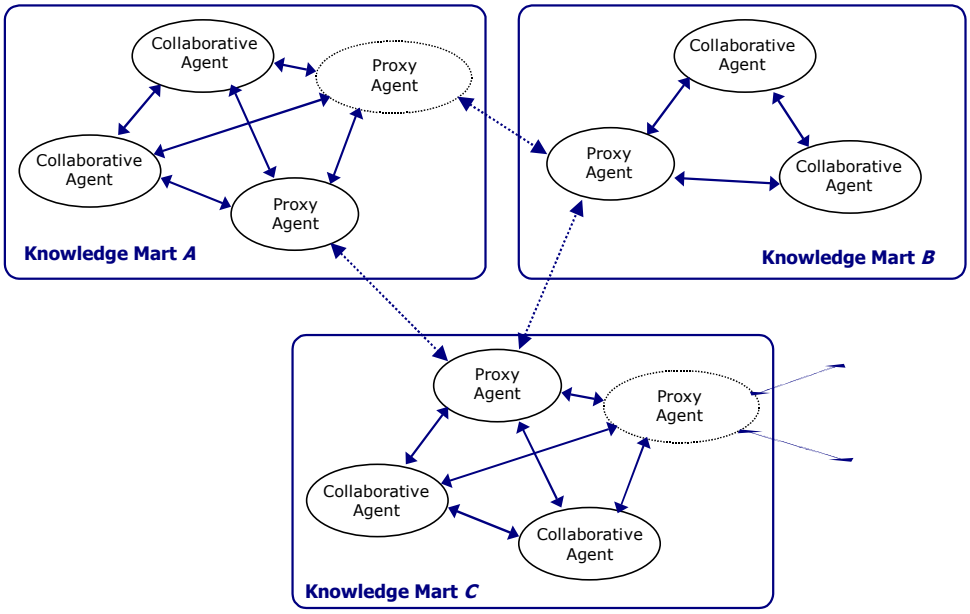


**Figure 1: Participative knowledge marts**

- A **Participative Knowledge Mart** (PKM) is a distributed group of agents that is trying to produce a piece of knowledge in a given domain.
- A **Participative Knowledge Warehouse** (PKW) is the place where knowledge produced in foreign marts is merged in a structured fashion.

Two or more PKMs can interact using representatives in a common PKW. When knowledge produced in a mart can affect performance in some other domain, a special proxy agent can act as representative in the foreign mart, according to the *proxy* design pattern (Gamma et al., 1994), so that interaction between marts is not tightly coupled.

## *Dynamics of Marts*

The architecture of heterogeneous knowledge domains into marts presents a number of questions and problems:

1. What knowledge domain are agents member of?
2. How is the corpus of knowledge built in a specific mart, how it is accepted by every agent and how it can evolve?
3. How are the domains initially spawned, after setting up a group of agents whose generated knowledge is not known in advance?
4. What would it happen if an agent changes the kind of knowledge that it produces, and this is better classified in another mart?

5. As time progresses, can knowledge that is produced in a mart be biased towards a different category?

Bringing forward the answers to these questions, it seems reasonable to dynamically establish the membership of agents into the marts. As well, division and/or fusion of marts can be needed to better reflect the knowledge-directed proposed structure. To represent the differences between knowledge marts, we define a *cognitive distance* between two agents as a measure of the similarity between the knowledge produced by both agents. The cognitive distance can be also defined between two marts, in the sense that these are dynamically formed groups of knowledge-carrying agents. In that case, clustering techniques can be readily applied to solve the above-mentioned problem of dynamic membership of agents into marts. Data about the cognitive distance between marts can be taken from agents' activity logs. For instance, a web server log file is a rich data source to determine a cognitive distance.

The following section introduces the dynamics of a participative multi-agent architecture, where agents in a mart try to convince each other to accept a given knowledge in some domain, so approaching the problem of building the corpus of knowledge. The aim is to allow agents to *consolidate* knowledge that is continuously produced in a PKM. Knowledge consolidation in a PKM is the establishment of knowledge as accepted by every agent operating in that PKM. Agents can reach a consensus on the PKM-wide accepted knowledge by the exchange of messages.

## *Agent Interaction in a Knowledge Mart*

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 interact is that agents can build and deliver proposals, which can be accepted or rejected. An example is the contract net protocol (Smith & Davis, 1981). The protocol is more sophisticated when recipients have a chance to build counterproposals that alter certain issues that were not satisfactory in the original proposal (Sierra et al., 1997). A more elaborated form of interaction allows parties to send justifications or arguments along with proposals. Such arguments indicate why proposals should be accepted (Parsons & Jennings, 1996; Sawamura & Maeda, 2000; Sycara, 1990).

Interaction between agents is carried out by exchanging proposals in a common language (Mayfield et al., 1995). Proposal interchange is directed by the goals and needs of participating agents. Although the formalisation of agents' communication language and goals is not included as an objective of our model, we assume a set of conventions about the language and protocol:

1. Agent rationality is modeled in terms of *preference relationships* or *relevance functions* (Fishburn, 1969), in order to allow agents to evaluate and compare proposals.
2. Relevant aspects of the interaction can be modeled as *issues* and *values* that change as the interaction progresses.
3. Agents deliberate and achieve an internal *state* and so record the history of interaction and their decisions.

An agent can be involved in several interaction processes. The protocol described below is used to carry out separate interaction processes where agents participate.

### Messages

The basic types of messages that can be exchanged between agents are the following:

- *proposal*($k$, $n$): Given an interaction process $n$, agents send a *proposal* message when they want a piece of knowledge $k$ to be consolidated in the PKM.
- *consolidate*($k$, $n$): Agents send a *consolidate* message when they reach a given state in the interaction protocol, and they want a previously submitted own proposal $k$ to be accepted in an interaction process $n$.

Sources and recipients of messages are not explicited as parameters in the messages described above. Addressing is concern of an underlying transport protocol that guarantees a reliable delivery. Moreover, message delivery to every agent in the same PKM has to be supported by some multicasting facility in the underlying transport.

### 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. Nevertheless, linguistic-expressed preferences (Herrera et al., 1996) can be integrated, as proposed by Delgado et al. (1998).

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

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

- Submitter's hierarchical level, useful when agents present different decision privileges in the PKM about the acceptance of proposals (e.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 are conducted by the training needs of the organization.
- Timestamp of the moment when a proposal was firstly submitted in the PKM (normally considered in the last case, when no other attribute decides).

**Proposal relevance**: The relevance of a proposal is defined as the set of proposal attributes considered when interacting.

**Proposal relevance function**: The relevance function $u(k)$ of a proposal $k$ in a PKM 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 PKM, denoted as $k_1 > k_2$, if $u(k_1) > u(k_2)$.

## Interaction Protocol in a PKM

Let $A_M = \{A_1, ..., A_n\}$ be a discrete set of agents, participating in a knowledge mart $M$.

**Start**: When $A_i$ wants a knowledge piece $k$ to be consolidated in $M$, it sends a *proposal*($k_i$, $n$) to every agent in $M$, initiating a new interaction process $n$. 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 interaction process $n$.

**Rule 1**: 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 tries to ratify it, by sending a *consolidate*($k_i$, $n$) to every agent in $M$. At the same time, $A_i$ starts a new timeout $t_1$.

**Rule 2**: When $A_i$ receives a *proposal*($k_j$, $n$) message from other agent $A_j$, referring to the same interaction process $n$, $A_i$ evaluates the new proposal $k_j$. If $k_i < k_j$, then $A_i$ sets a new timeout $t_1$, waiting for proposal $k_j$ to be ratified. Then, $A_i$ proceeds as follows:

2.1. If $A_i$ does not receive any proposal referred to interaction process $n$ before $t_1$ expires, then $A_i$ initiates back the protocol with the same proposal $k_i$.

2.2. If $A_i$ receives a *consolidate*($k_j$, $n$), with $k_j > k_i$, for $j \neq i$, before $t_1$ expires, and referring to the same interaction process $n$, then $A_i$ gives up the initial proposal and the protocol finishes unsuccessfully.

2.3. If $A_i$ receives a new *proposal*($k_j$, $n$), with $k_i < k_j$, it extends the timeout $t_1$.

We are considering two different timeouts over the course of the protocol, one for each phase that can be noticed in the interaction. Timeout $t_0$ is used for the distribution phase, that occurs after an agent submits a proposal. Timeout $t_1$ is used for the consolidation phase, that occurs if there is a proposal waiting to be consolidated (this can occur whether $t_0$ expired or a $t_0$-waiting agent received a proposal that was evaluated as preferred).

At 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.

An agent $A_i$ can participate in several interaction processes. Each interaction process is handled separately, by initiating a new execution thread of the protocol.

## *Activation Events*

In any moment, an agent $A_i$ can be involved in an interaction process $n$ due to the arrival of a message from $A_j$ referred to $n$. The following rules describe the actions to undertake by agent $A_i$ 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 $A_i$ had sent a *proposal*($k_i$, $n$) and is waiting for $t_0$ timeout, then it can perform one of the following actions:
        - If $k_j > k_i$, then $A_i$ starts timeout $t_1$ and it keeps waiting for proposal $k_j$ to be ratified or for an alternative proposal to come.
        - If $k_j < k_i$, then $A_i$ sends again its last $n$-related proposal back to $A_j$, and extends $t_0$ timeout.
    - **Rule B**: If $A_i$ had sent a *proposal*($k_i$, $n$) and is waiting for $t_1$ timeout, then it can perform one of the following actions:
        - If $k_j > k_i$, then $A_i$ acts in the same manner as in rule 2.3 and it keeps waiting for proposal $k_j$ to be ratified.
        - If $k_j < k_i$, then $A_i$ sends last proposal it sent in the interaction process back to $A_j$, and extends $t_1$ timeout.

- **Rule C**: If $A_i$ had not sent any message referred to the same interaction process $n$, it does nothing.

- If $A_i$ receives a *consolidate*($k_j$, $n$) from $A_j$:
  - **Rule A**: If $A_i$ had sent a *proposal*($k_i$, $n$) and is waiting for $t_0$ timeout, then it can perform one of the following actions:
    - If $k_j > k_i$, then $A_i$ acts in the same manner as in rule 2.2 and the protocol finishes unsuccessfully.
    - If $k_j < k_i$, then $A_i$ sends again its last proposal back to $A_j$, and extends $t_0$ timeout.
  - **Rule B**: If $A_i$ had sent a *proposal*($k_i$, $n$) and is waiting for $t_1$ timeout, then it can perform one of the following actions:
    - If $k_j > k_i$, then $A_i$ acts in the same manner as in rule 2.2 and the protocol finishes unsuccessfully.
    - If $k_j < k_i$, then $A_i$ acts in the same manner as in rule 2.1 and initiates back the protocol with the same proposal.
  - **Rule C**: If $A_i$ had not sent any message referred to the same interaction process $n$, it does nothing.

### *Protocol Variants*

The interaction protocol described above uses two message types (i.e., proposal and consolidate), but some variants using additional message types to express different semantics can also be formulated:

- *retract*($k$, $n$): Agents can retract from a previous proposal $k$ by issuing this message referred to an interaction process $n$.
- *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 formulating and issuing a new proposal.

These types of message can speed up the development of the protocol, but they are not completely necessary for the success of the process.

## Case Study: Participative Development of Learning Objects

Recently, organizations and companies have developed initiatives for learning resources standardization, as LALO (Learning Architectures and Learning Objects) (CEdMA, 2000) and SCORM (Sharable Content Object Reference Model) (ADL, 2001). Most of them are based upon specifications of IMS (Instructional Management Systems) standards. The IMS Global Learning Consortium has recently released the final release, version 1.1.2 of the Content Packaging Specification (IMS, 2001), which provides the functionality to describe and package learning materials, such as an individual course or a collection of courses, into interoperable, distributable packages.

The IMS specification defines the structure of a learning object by means of an XML *manifest* file, as depicted in figure 2. Proposed additions or modifications are transformed into changes to the manifest file. When constructing a learning object, the builder can use an editing tool —for instance, *Microsoft LRN Toolkit* (Microsoft, 2002) or ToolBook II (Click2learn, 2001)— that provide an implementation of the IMS Content Packaging Specification. Participative development of learning objects is not supported

in such tools, which present a unipersonal vision of creation, edition, viewing, and testing of learning objects.

In the participative development of a learning object, several educational designers may wish to contribute, making some modification in the structure of the course, or adding some object to the course contents. Two or more authors can cooperate through 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.

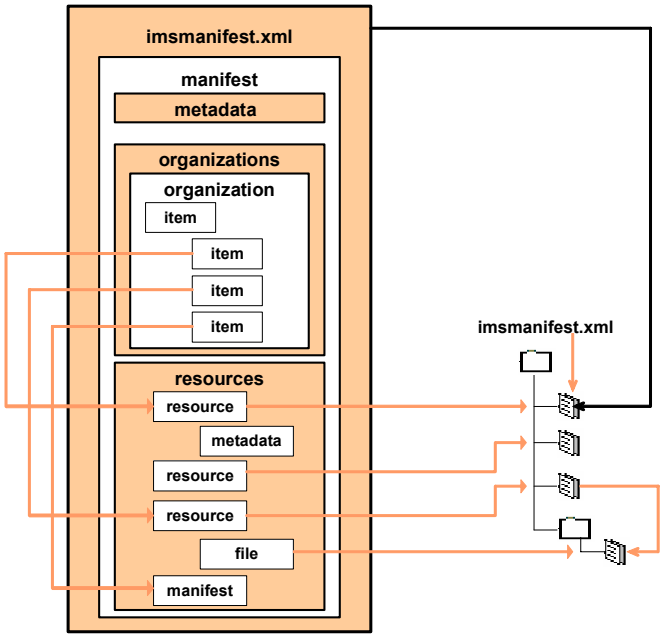


**Figure 2: IMS manifest file structure, © Microsoft Corp.**

## *Issues within Participative Development*

The participative development of learning resources becomes harder than if carried out in a centralized environment. Firstly, in a highly distributed environment —for instance, the faculty staff in a virtual university—, the holding of synchronous —physical or virtual— meetings is not frequent, since the exchange of ideas between members of the distributed workgroup is an asynchronous process, where participants may keep their own pace within the interchange.

In a second stage, developers of learning objects make specific use of their knowledge on a given discipline. For example, when some lecturers of a faculty staff make up a syllabus, they use their knowledge on a group of related subjects. The determination of responsibilities within the participative development is critical to the quality of the learning products.

Moreover, the processes accomplished in the collaboration are not isolated from one another, but they keep inter-dependencies. It is desirable that these dependencies could be eliminated or reduced. The intention of the proposed architecture is to alleviate these issues.

## *A Running Example*

Let us consider the development of a learning object named “Introduction to XML”. In a given moment, an author $A_1$ knows the currently consolidated object and elaborates a proposal $p$ (see figure 3). In the meantime, another author $A_2$ elaborates a new proposal $q$ (see figure 4). There may be several processes initiated, each one affecting a section of the learning object that can be negotiated separately (e.g, `tableofcontents`, `resources`, and `metadata`). We will restrict our example to the `tableofcontents`

structure of the `organizations` section in the manifest file, and assume that the interaction process *n* is devoted to the `tableofcontents` section. Nevertheless, participative development can be also extended to `resources` or `metadata` sections.

When authors submit proposals, they will include the differences between both files, and referred to the `tableofcontents` interaction. The rest of collaborating authors receive and evaluate the proposal, according to a set of previously agreed criteria, like those described above. Then, the interaction 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. Although authors' behavior is an asynchronous process, agents interaction protocol helps to synchronize their operations.

```
<?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: Proposal *p* for the manifest file while developing a learning object**

In our example, the relevance of a proposal can be graded by the fulfillment of a set of instructional outcomes from trainees within the following objectives:

1. Ability to program XHTML (i.e., XML-generated HTML) web applications
2. Ability to program server-side web applications
3. Ability to program XML data exchange applications

The sequence of events spawned by the execution of the protocol by every agent is depicted in figure 5. The interaction begins when agents $A_1$ and $A_2$ submit proposals *p* and *q* respectively.

```
<?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: Proposal *q* for the manifest file while developing a learning object**

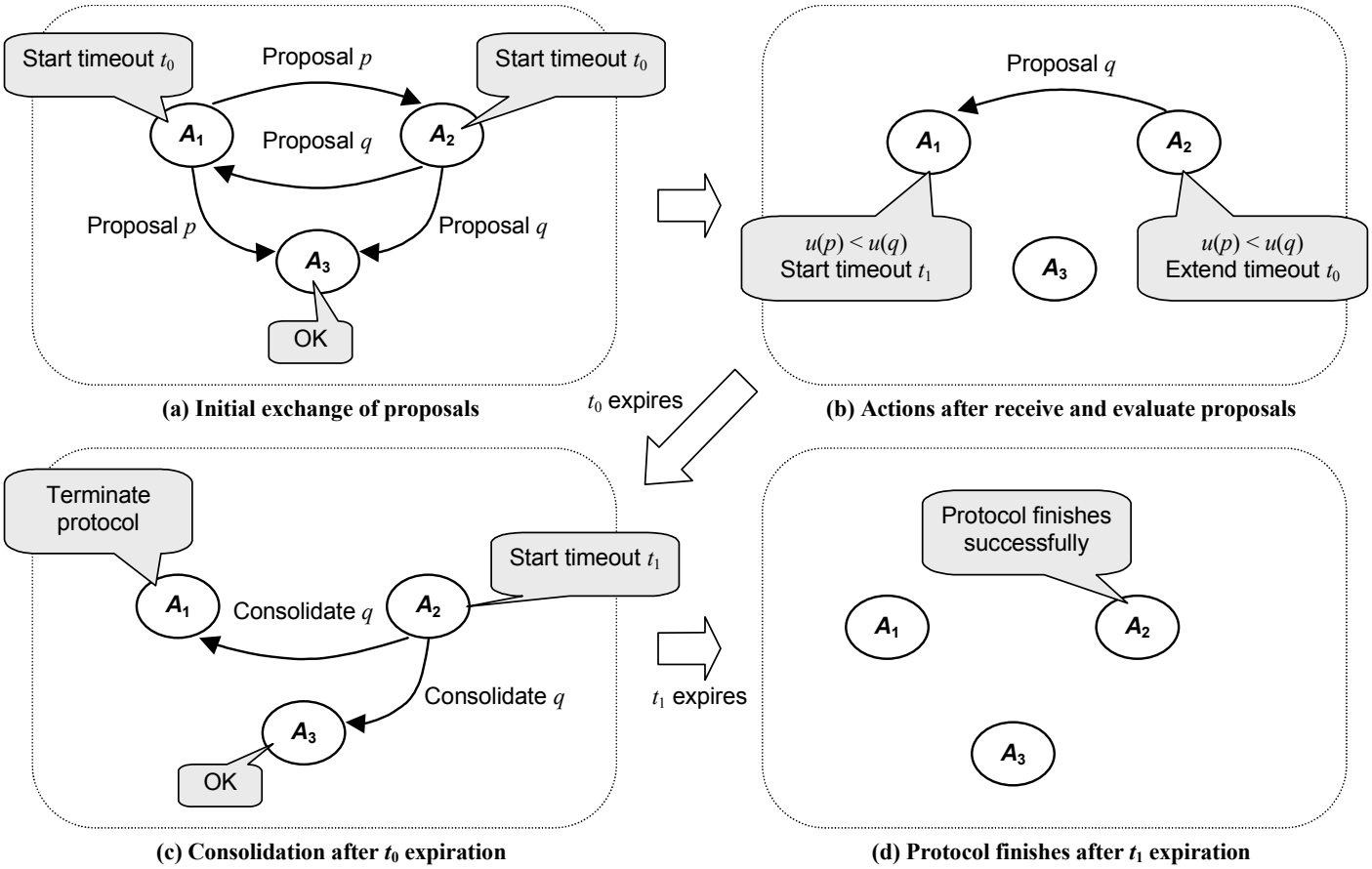


**Figure 5: Execution example of the interaction protocol**

(a) Both $A_1$ and $A_2$ receive each other's proposal and begin the distribution phase, so starting timeout $t_0$. Proposals $p$ and $q$ also arrive to $A_3$, which is not participating in the process and silently receives them.

(b) $A_1$ compares $q$ to $p$, turning out that its proposal has a worse evaluation. It is reasonable that an evaluation of proposal $p$ obtains a higher value than $q$, as for the second objective described above. Concerning first and third objectives, any relevance function should outcome similar values for both proposals, so they would not be decisive. Then, $A_1$ starts timeout $t_1$, giving $q$ a chance to be consolidated. On the other hand, $A_2$ also compares both proposals and reminds $A_1$ of the results by sending again $q$, then extending timeout $t_0$ in order to give a chance for other agents' proposals to come.

(c) When timeout $t_0$ expires, $A_2$ sends a consolidation message for $q$ that arrives to every agent in the mart. At the reception, $A_1$ finishes the protocol because it is expecting the consolidation for $q$. $A_3$ simply accepts the notification.

(d) Finally, at the expiration of $t_1$, $A_2$ is confirmed about the end of the consolidation phase for $q$ and its execution of the protocol finishes successfully. Therefore, every agent in the mart will eventually know about the consolidation of the proposal.

### *A Multiple-mart Architectural Model*

The situation previously described leads us to the hypothesis that a group of agents can help instructional designers to jointly, asynchronously develop learning objects, by coordinating their creation activities. Different agents can act as representatives of instructional designers, according to the following principles:

- Agents can be structured into separable knowledge domains of interaction, so reflecting the knowledge differences between instructional designers.
- A dynamic re-structuring of the knowledge domains can help to reduce the inter-dependencies that can occur within the disciplines afforded by instructional designers.

Participative development of the 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 can be helpful to structure the interaction.

The interaction protocol deals with the asynchronous exchange among learning objects developers by synchronizing the interaction among their representative agents. The learning objects that are generated are progressively consolidated in domains that are concerned with the knowledge that is under development. The inclusion of some content in a given course is under the responsibility of a lecturer, thus reflecting a chain-of-responsibility in the development.

As well, developers' specific knowledge domains are well reflected in the multi-domain structuring of agents into knowledge marts. Moreover, inter-dependencies are reduced by the dynamic set-up of marts.

## Conclusions

This work presents a model to develop a participative multi-agent architecture, suggesting its mapping to the participative development of learning objects. Multi-agent interaction protocols can be developed according to top-down or bottom-up approaches. Bottom-up approaches sector agents into territories and duplicate the social-level knowledge in each territory, which can lead to the problem of inconsistency. The architecture presented here is a bottom-up approach to the design of participative multi-agent sys-

tems, where every PKM holds responsibilities on some domain-level knowledge, while cooperation-level knowledge interfaces to other domains are well-defined. This structuring of knowledge marts can help to reduce inconsistencies between agent territories.

The participative 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. We are also conducting tests on the impact of the number of agents in the overall effectiveness of the model.

### *Improvements and Future Work*

For the sake of simplicity, we defined a two-tier architecture. Nevertheless, it can be easily extended to a multiple-tier architecture, where agents interact in lower-level domains to consolidate some knowledge, before they try to interact in higher-level domains.

On another hand, interaction processes are not completely independent from one another. Therefore, the participation of agents in the marts can be dynamic, such that an agent can change its membership to some other mart, if the knowledge produced by the agent affects interaction processes carried out in that mart. Since 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 PKMs are proposed to be dependent on agents' ontology-based expressed interests.

## References

ADL (2001). Sharable Content Object Reference Model, version 1.1. Advanced Distributed Learning. Retrieved January 15, 2001 from the World Wide Web: http://www.adlnet.org/scorm/docs/scorm1.1.pdf

ALT (1996). Definition of Learning Technology. Association for Learning Technology. Retrieved August 14, 2001 from the World Wide Web http://www.warwick.ac.uk/alt-E/rolling/discussion/69

Bateson, G. (1972). *Steps to an Ecology of Mind.* London: International Textbook Publishing.

CEdMA (2001). Learning Architectures and Learning Objects. Computer Education Management Association. Retrieved May 15, 2001 from The World Wide Web http://www.cedma.org/guestlalo.html.

Click2learn (2001). Toolbook authoring products. Click2learn, Inc. Retrieved March 10, 2002 from the World Wide Web http://home.click2learn.com/en/toolbook/index.asp.

Delgado, M., Herrera, F., Herrera-Viedma, E., & Martínez, L. (1998). Combining Numerical and Linguistic Information in Group Decision Making. *Information Sciences*, 7, 177–194.

Farance, F., & Tonkel, J. (2001). Learning Technology Systems Architecture (LTSA) Draft 8. Technical report. IEEE Learning Technology Standards Committee (LTSC). Retrieved April 6, 2001 from The World Wide Web http://ltsc.ieee.org/doc/wg1/IEEE_1484_01_D01_LTSA.pdf.

Finin, T., Fritzson, R., McKay, D., & McEntire, R. (1994). KQML as an Agent Communication Language, In *Proceedings of the 3rd Int. Conf. on Information and Knowledge Management*, pp. 456–463, Gaithersburg, Maryland. ACM Press.

Fishburn, P. C. (1969). *Utility Theory for Decision Making.* Huntington, NY: Robert E. Krieger Publishing Company.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns.* Reading, Massachusetts: Addison-Wesley Publishing Company.

Herrera, F., Herrera-Viedma, E., & Verdegay, J. (1996). A Linguistic Decision Process in Group Decision Making. *Group Decision and Negotiation*, 5, 165–176.

IMS (2001). IMS Content Packaging Specification. Instructional Management Systems. Technical report. IMS Global Learning Consortium. Retrieved April 6, 2001 from The World Wide Web http://www.imsproject.org/content/packaging/index.html.

Jennings, N. R., & Campos, J. R. (1997). Towards a Social Level Characterisation of Socially Responsible Agents. *IEEE Proceedings on Software Engineering*, 144(1), 11–25.

Maes, P. (1994). Agents that Reduce Work and Information Overload. *Communications of the ACM*, 37(7), 31–40.

Mayfield, J., Labrou, Y., & Finin, T. (1995). Desiderata for Agent Communication Languages. In *AAAI Spring Symposium on Information Gathering*.

Microsoft (2002). LRN 3.0 Toolkit, Microsoft Corporation. Retrieved March 10, 2002 from the World Wide Web http://www.microsoft.com/elearn/support.asp.

Parsons, S. D., & Jennings, N. R. (1996). Negotiation Through Argumentation - A Preliminary Report. In *Proc. Second Int. Conf. on Multi-Agent Systems*, pages 267–274, Kyoto, Japan.

Sawamura, H., & Maeda, S. (2000). An Argumentation-based Model of Multi-Agent Systems. In H. Jaakkola and H. Kangassalo (eds.), *Proceedings of the 10th European-Japanese Conference on Information Modelling and Knowledge Bases*, pp. 96–109, Saariselkä, Finland.

Sierra, C., Faratin, P., & Jennings, N. R. (1997). A Service-Oriented Negotiation Model between Autonomous Agents. In *Proc. 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, pp. 17–35, Ronneby, Sweden.

Smith, R. G. (1980). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, 12, 1104–1113.

Smith, R. G., & Davis, R. (1981). Frameworks for Cooperation in Distributed Problem Solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1), 61–70.

Swanstrom, E. (1999). *Knowledge Management: Modeling and Managing the Knowledge Process.* John Wiley & Sons.

Sycara, K. (1990). Persuasive Argumentation in Negotiation. *Theory and Decision*, 28, 203–242.

Von Goldammer, E., & Kaehr, R. (1988). Poly-contextural Modeling of Heterarchy in Brain Function. In R. M. J. Cottrill (eds.), *Models of Brain Function*, pp. 463–497, Cambridge University Press.

Wiley, D. A. (2002). Connecting Learning Objects to Instructional Design Theory: A Definition, a Metaphor, and a Taxonomy. In D. A. Wiley (editor), *The Instructional Use of Learning Objects*, pages 3–23, Agency for Instructional Technology, Association for Educational Communications & Technology.

Yuan, S. T., & Wu, Z. L. (2000). An Infrastructure for Engineering Cooperative Agents. *International Journal of Software Engineering*, 10(6), 681–711.

# Biographies

**Juan-Manuel Dodero** received a degree in Computer Science and a Master degree in Knowledge Engineering from Universidad Politécnica de Madrid. He collaborated with the laboratory of Artificial Intelligence of the Facultad de Informática de Madrid (School of Computer Science of Madrid). Since 1994, he was a lecturer at the Computer Science School of the Universidad Pontificia de Salamanca en Madrid. He has also been a consultant on object technologies and knowledge management for several companies. Starting from 1999, he is a lecturer at the Universidad Carlos III de Madrid (Escuela Politécnica Superior). His interests mainly concern topics such as Cooperative Information Systems Architectures, Agent Technology, Knowledge Management and their applications to education.

**Ignacio Aedo** received a degree and a Ph.D. in Computer Science from Universidad Politécnica de Madrid. Starting from 1991, he is lecturer at the Universidad Carlos III de Madrid (Escuela Politécnica Superior). His interests mainly focus on topics such as hypermedia, media integration systems, electronic books, electronic document systems, development methodology and knowledge representation systems. In 1990, he started his research activity in the field of hypermedia systems, on which he is still involved.

**Paloma Díaz-Pérez** received a degree and a Ph.D. in Computer Science from Universidad Politécnica de Madrid. She collaborated with the laboratory of Software Engineering of the F.I.M. Starting from 1992, she is a lecturer at the Universidad Carlos III de Madrid (Escuela Politécnica Superior). Her interests mainly concern topics such as hypermedia, electronic document systems, CASE, software development methodology, and formal models for representing information.