Representation of Coordination Mechanisms in IMS LD

Representation of Coordination Mechanisms in IMS LD

Yongwu Miao *
Educational Technology Expertise Center,
Open University of The Netherlands,
Valkenburg 177, 6419 AT, Heerlen, The Netherlands
Tel: +31.45.5762797
Fax number: +31.45.5762800
Email: yongwu.miao@ou.nl

Daniel Burgos
Atos Research & Innovation,
Avda. Diagonal, 188, 3a Planta,
08018 Barcelona, Spain
Tel: +34 93 4861818
Fax: +34 93 4860766
Email: Daniel.burgos@atosresearch.eu

David Griffiths
The Institute for Educational Cybernetics
The University of Bolton
Deane Road, BL3 5AB, Bolton, UK
Tel: +44 1204 90 3660
Email: d.griffiths.1@gmail.com

Rob Koper
Educational Technology Expertise Center,
Open University of The Netherlands,
Valkenburg 177, 6419 AT, Heerlen, The Netherlands
Tel: +31.45.57622657
Fax number: +31.45.5762800
Email: rob.koper@ou.nl

Representation of Coordination Mechanisms in IMS LD

ABSTRACT

Group interaction has to be meticulously designed to foster effective and efficient collaborative learning. The IMS Learning Design specification (IMS LD) can be used to create a formal representation of group interaction and the model can then be used to scaffold group interaction by means of coordination support at run-time. In this chapter, we investigate the expressiveness of IMS LD in representing coordination mechanisms by using coordination theory as an analytical framework. We have found that IMS LD can represent almost all the basic coordination mechanisms. We have also identified some hurdles to be overcome in representing certain coordination mechanisms. According to coordination theory, common coordination mechanisms can be reused in different settings. We briefly explore the feasibility of representing coordination mechanisms at a high-level of abstraction, which will be easier for instruction designers and teachers to understand and use.

INTRODUCTION

Group-based learning is an instructional strategy that provides a group of learners with intensive group interaction that can deepen individual learners' understanding. Well-organized group-based learning may result in collaboratively produced knowledge objects or conceptual artifacts which could not be created by any individual learner in the group acting alone. However, the benefits of this instructional strategy have a cost, because additional coordination activities have to be carried out while learners perform learning activities. Examples of such coordination activities are allocating tasks, distributing and exchanging information, and managing work sequences. Although coordination activities do not directly contribute to the production of knowledge objects or conceptual artifacts, they have an influence on the effectiveness and efficiency of group-based learning, and sometimes on its success or failure.

In face-to-face learning rich communication channels are available to support group interaction. These are lost in computer-based learning, and so in this environment there is a need to provide computational coordination mechanisms. One promising technical solution is to provide a formal model of a well-designed group interaction by using a process modeling language, and then to coordinate learners' interactions according to this model in a language-compatible execution environment. This enables learners to focus on learning activities without having to pay too much attention to coordination problems, and so supports enhanced effectiveness and efficiency of group-based learning in computer-based environments.

IMS Learning Design (IMSLD 2003) is an educational process modeling language which can be used to model a wider range of pedagogical strategies, including collaborative learning (Koper and Olivier 2004). A basic introduction to IMS LD is available in the chapter (Using the IMS LD Standard to Describe Learning Designs, Koper and Miao, this book). The purpose of this present chapter is to systematically investigate the expressiveness of IMS LD in representing coordination mechanisms which support group interaction, and the approach taken is to use coordination theory as an analytical framework. We also provide XML (Extensible Markup Language) code, to illustrate how group interaction can be represented in IMS LD.

It is important to note that characteristics of group-based learning processes vary from well-structured to highly fluid. Highly fluid collaborative processes, in which it is unpredictable who will take which action when, and how other group members will respond, are not well suited to coordination using computational mechanisms. The attempt to specify a fluid collaborative process in detail often raises the so-called "over-scripting" problem (Dillenbourg, 2002), which may restrict group interaction to some extent. Some fluid collaborations are suited to coordination by human users. These may be defined in IMS LD, for example, as a collaborative activity with a conference service (e.g., an audio/video conferencing, text-based chat tool, or a discussion forum). The users (e.g., tutors and students) are expected to solve their coordination problems by using functions offered by the service. It may be seen that using this approach the coordination within an activity is not specified at the process level in the learning design, and that responsibility for process control is shifted to the user at execution time. This is, therefore, outside the scope of this chapter, which focuses on how computational mechanisms can be represented in IMS LD.

BACKGROUND

This section briefly introduces group-based learning and coordination theory.

Group-based Learning and Collaboration Scripts

Learning in small groups has been intensively researched since the 1970s. According to Tribe (1994) there are two main types of purpose for group-based learning in higher education: those related to skills acquisition and those related to academic aims. As Tribe (1994) summarized, the skills acquired in group-based learning cover such interpersonal competences as oral communication, active listening, group leadership, group membership, the ability to examine assumptions, and the ability to tolerate ambiguities. All of these skills are highly valued in employment. The academic objectives which build on these employment skills include the ability to understand a text, question a line of argument, follow up a lecture, and gauge an individual's progress on a particular course or evaluate a course.

According to (Strijbos & Martens 2001, Strijbos, Martens, et. al. 2004) there is agreement on five components of 'group-based learning'. As Strijbos el. al. summarize, firstly,

groups are composed of either a minimum of two up to six participants. Secondly, group-based learning is characterized by 'positive interdependence', which refers to the degree to which the performance of a single member is dependent on the performance of all others (Johnson, 1981). A third component is the task, which must be a genuine group task, in which the effort of all group members is needed. A fourth component is 'individual accountability'. This refers to each student's individual responsibility for a specific aspect of the group process or group performance (or both). Individual accountability is enhanced through grading students for their individual effort or performance, as well as the group's performance. The fifth and final component is a shift from 'teacher centered' to 'student centered'.

Early studies on group-based learning focused on the role of independent variables that might influence the learning outcome, e.g. group size and group dynamics. Recent studies, however, analyse group interactions in order to ground the design of the support to be provided. According to Dillenbourg (1999), the key to understanding collaborative learning is to gain an understanding of the interactions among individuals. Recently in the Computer-supported collaborative learning (CSCL) community, the design of collaboration scripts has been a new focus area. The basic idea is to formally describe group interaction by using a scripting language and then to coordinate group members and their actions by executing collaboration scripts (O'Donnell and Dansereau 1992; Dillenbourg 2002; Kollar, Fischer et al. 2005; Miao, Hoeksema et al. 2005; Weinberger, Stegmann et al. 2005, Fischer, et. al. 2007). Some efforts (e.g., Caeiro et al. 2003; Hernandez et al. 2004; Miao, Hoeksema et al. 2005; Van Es and Koper 2006) have been made to investigate whether IMS LD is sufficiently expressive to represent collaborative learning processes effectively, usually by analyzing special cases. The most serious research in this direction was done by Van Es and Koper (2006), which investigated many examples, randomly selected from 6034 lesson plans. In the research described in this chapter, not only a case study method (the case used here is mainly for the purpose of explanation), but also a theory-based analysis method is adopted to systematically test the capacity of IMS LD in representing coordination mechanisms.

Coordination Theory

Coordination theory concerns the interdisciplinary study of coordination, which is defined as the process of managing dependencies between activities. Malone and Crowston (1994) analyzed processes in terms of actors performing interdependent tasks. These tasks might require or create resources of various types. Coordination theory provides a theoretical framework for analyzing coordination in complex processes, thus contributing to user task analysis and modeling. It has been applied in many fields, including computer science, organization theory, economics, management science, sociology, social psychology, anthropology, linguistics, law, political science, and so on. The research reported here is the first time that coordination theory has been applied to education.

One of the most powerful contributions of coordination theory is to systematically identify and analyze a wide variety of *dependencies*. Three elementary dependency types are identified in coordination theory: 1. *Sharing*, 2. *Flow*, and 3. *Fit*. In sharing dependencies two or more activities share the same resource(s). *Sharing dependency* frequently occurs when one resource is used by a number of people or activities, whether that resource is a machine on a factory floor, a budget, or a room, or anything else which is used in multiple activities. In *flow dependencies* resources produced by one activity are consumed by one or more subsequent activities. The concept of flow is intuitive and ubiquitous, emerging from the succession of events in human activity. In *fit dependencies* two activities concurrently produce the components of the same resource, and these have to fit together. A good example of *fit* is the design of a car, where one engineer designs the engine, another designs the body, and so forth. Dependencies arise between the activities because all the parts have to fit together in the same car.

It is important to note that these three dependency types can be further specialized. For example, the flow dependency can be divided into three sub-dependencies: *precedence*, *transfer* and *usability*. *Precedence dependency* indicates that the actor performing the second task has to know when the resource is available and the task can be started. *Transfer dependency* indicates that the resource must be moved from the activity in which it was created to the activity in which it is consumed. Finally, *usability dependency* indicates that the resource created by the first task must be appropriate for the needs of the second task. The fit dependency can be further specified as a *decomposition dependency* between task and sub-task.

According to coordination theory, all dependencies in any relationship can be analyzed as either combinations of, or more specialized types of, these three elementary types or their sub-types. The theory describes how these dependencies can present actors in organizations with *coordination problems* which constrain the efficiency of task performance. To overcome coordination problems, actors must perform additional activities such as allocating tasks and control workflow and information-flow, which Malone and Crowston called *coordination mechanisms* or *coordination activities* (Malone and Crowston, 1994). Many such mechanisms to manage dependencies have been identified in organizations. Different organizations which have similar goals and achieve them using more or less the same set of coordination activities will have to manage the same dependencies. Nevertheless, they may choose to use different coordination mechanisms, thus resulting in different processes (Crowston and Osborn 1998). The best process to use depends on situational factors and often involves trade offs.

REPRESENTATION OF COORDINATION MECHANISMS IN IMS LD: A CASE STUDY OF GROUP_BASED LEARNING

In this section, based on coordination theory, we analyze the coordination problems which arise in group-based learning processes, and also systematically explore the degree to which IMS LD can represent possible coordination mechanisms for supporting group interaction, either directly or indirectly. The investigation is conducted and explained using the "Knowledge Convergence Script" use case, which is briefly introduced at the beginning of this section.

Knowledge Convergence Script

We have chosen to model an example of group-based learning which is well documented in the literature (Weinberger, Fischer et al. 2004). This was conducted in a web-based environment, with a small group of three learners who were required to write three reports about three cases. Following the original design the whole process is carried out in four stages.

1. Case reporting: Each learner reads a different case and writes a report about the case which they have read. When all three learners have finished their reports, they pass them on to designated co-learners the first round of a pre-defined pattern of rotation.
2. Criticizing 1: Each learner comments on the report which they have received. When all three have finished the first round of comments, they rotate the reports again, together with the first round comments.
3. Criticizing 2: Each learner comments on the newly transferred report and the associated comment. When all three have finished the second round of comments, they rotate the reports again, together with the first and second round comments.
4. Finalizing the report: Each report returns to the original author together with two comments. Each learner revises their own report (writes a synthesis to merge the ideas of other learners) in the light of their comments.

The "Knowledge Convergence Script" has been implemented in a web-based collaborative learning environment, and it is reported that this group-based learning strategy is effective and efficient (Weinberger, Fischer et al. 2004). In supporting this group-based learning strategy we use process modeling and execution approach, rather than a software development approach. Figure 1 illustrates the process model, using the following conventions:

- light-gray rectangles represent stages
- dark-gray rectangles represent activities
- white rectangles represent artifacts
- solid arrows indicates workflows
- dashed arrows indicate information-flows.

Three learners are shown: *learner1*, *learner2*, and *learner3*, who work through a four-stage work procedure including *Case reporting*, *Criticizing 1*, *Criticizing 2*, and *Finalizing the report*. At each stage, three learners perform activities in parallel to produce artifacts which will be used as input of succeeding activities carried out by their peers. For example, at the first stage *learner1* performs activity *reporting1*. He/she reads

Case1 and produces artifact *InitialReport1*, which is then transferred to the activity *criticizing2-1* at the second stage. *Learner2* produces artifact *comment2-1*, which is then transferred together with *Case1* and *InitialReport1* to *learner3*. At the third stage *learner3* reads *Case1*, *InitialReport1*, and *Comment2-1* and writes *Comment3-1*. Finally, all documents associated with *Report1* are transferred to *learner1*. He/she improves *Report1* based on the received comments, and then produces a final version of the case report *FinalReport1*.

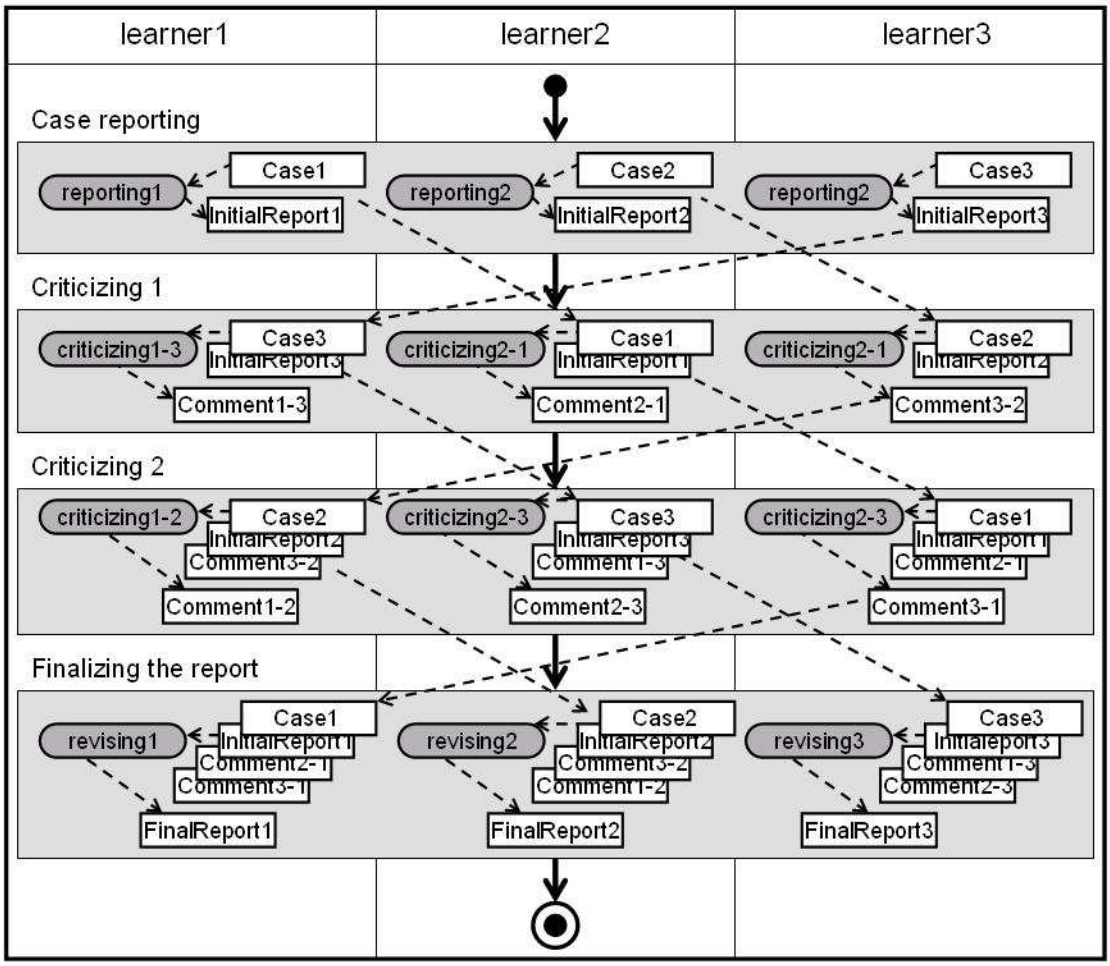

Figure 1. An activity diagram of "Knowledge Convergence Script".

We use IMS LD to specify this strategy in the form of XML. The resulting model (KCS uol, 2007) can be executed in any IMS LD compliant run-time environment, such as CopperCore (Vogten, et al., 2006). Figure 2 shows a screenshot of CopperCore used to run this script when *learner1* is writing the final report. The top-left pane shows the work procedure of the user. The bottom-left part shows all *environments* associated with the activity currently being performed, which include the documents to be accessed by the user. When the user clicks a learning object (such as a case and a comment made available in the environment), the content of the learning object is presented in the right

part of the window. In the screen shown below the user has selected the final activity *write final report*. The main area of the window presents the activity-description of *write final report* activity, in which the user writes the final version of his/her case report as shown in figure 2. It is important to note that the main goal of this research is not to study whether this group-based learning strategy is effective or efficient, but to investigate and demonstrate the expressiveness of IMS LD in modeling group-based learning strategy. Moreover, we observe that various group-based learning strategies can be adopted to achieve the same learning goal, and that no single strategy is ideal for all situations. Accordingly we designed some alternatives, which are not intended to improve this group-based learning process, but rather to provide the basis for a discussion of possible coordination mechanisms.

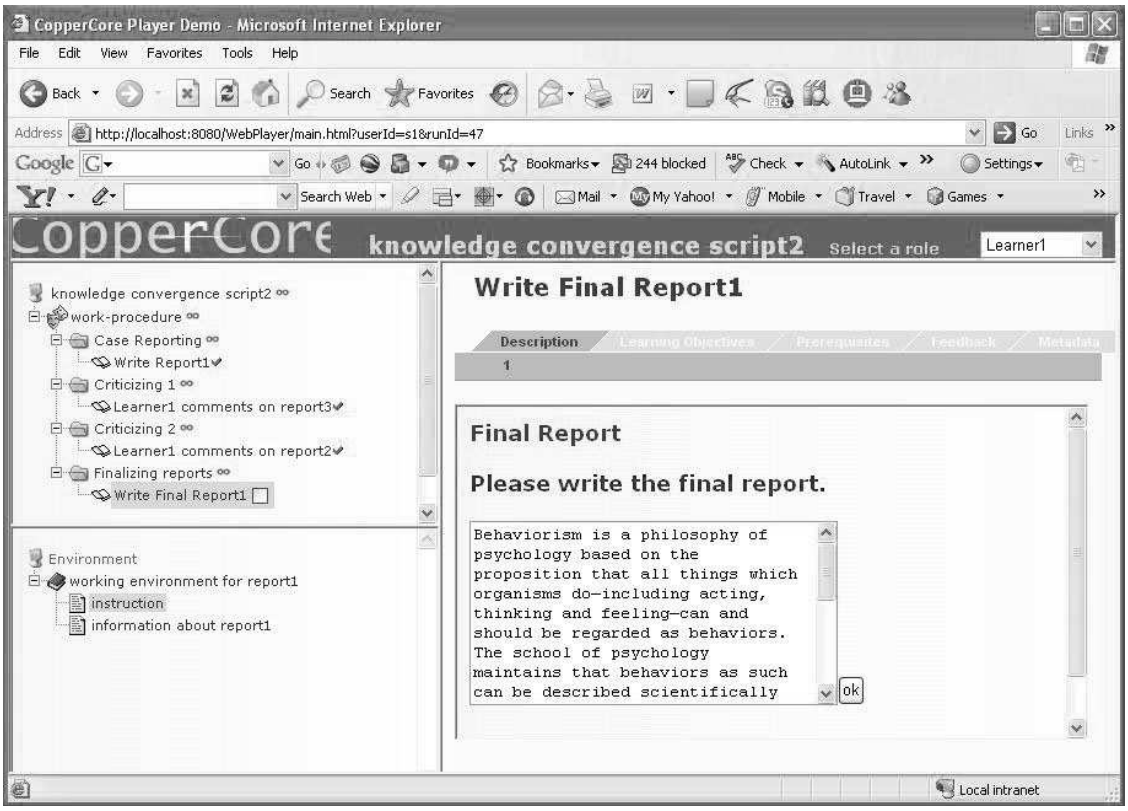

Figure 2. A screenshot of CopperCore running the "Knowledge Convergence Script".

Analyses of dependencies and possible coordination mechanisms in group-based learning processes

In this section, we investigate various forms of dependences in group-based learning processes from the perspective of coordination theory. The "Knowledge Convergence Script" and its alternatives are used as examples to analyze and explain the coordination mechanisms to manage various dependencies in group-based learning processes.

Sharing dependencies

In the activities carried out in this use case each learner has to read three cases and make contributions to each report. Thus the resources shared by the activities are three learners and three cases. If learners work without any coordination mechanism for managing the sharing dependencies described above, the result will be disorder, with each learner performing any task at any time.

The coordination mechanism used in the original design is to pre-define the allocation of learners and cases to activities, so that some are carried out concurrently, and others at different times. In order to support this coordination mechanism it is necessary to represent the bindings between the actors and the activities which they will carry out, either concurrently or at different times. It is also necessary to ensure that actors have access to the appropriate cases when they have to carry out a particular activity. Another possible coordination mechanism for managing sharing dependencies is that three learners and one case will be allocated to individual activities in turn. Each activity is itself a collaborative task. In order to support this coordination mechanism, it is necessary to represent the binding between the multiple actors and the same activity at the same time, and to represent the use of the communication tools used to exchange their ideas and create the report.

These two strategies are static coordination mechanisms, which manage the sharing dependencies in a pre-defined manner. If it is not decided in advance which learner will be responsible for reporting on which case then a dynamic coordination process will be required which responds to the dynamics of the learning process. An example of a dynamic coordination mechanism is “first come, first served”, and this mechanism can be applied to determining the pattern of rotation. For example, we could add a register activity for each role, but not allocate any activity to any role in design time. At run-time, three users will register to carry out the process, and according to the sequence of their registrations, the activities will be allocated and the artifacts rotated.

Flow dependencies

As mentioned before, the flow dependency has three sub-dependencies: *1) precedence*, *2) transfer* and 3) *usability*. We now analyze these types of dependencies in the group-based learning.

1) Precedence: In the use case, there are precedence dependencies between some activities. For example, when one learner has finished the activity of creating an initial report, the other two can comment on it in turn. Only after other two learners provide their comments can the first learner write a synthesis.

Normally, the coordination mechanism used to manage precedence dependencies is event-driven. This means that an event (e.g., the termination of an activity and the available of a resource) triggers the start the succeeding activity. In complex learning

process, branching, forking, and joining are possible coordination mechanisms. Branching means a control that only one succeeding activity will be triggered among several candidates according to a condition. Forking refers to the control that two or more succeeding activities will start in parallel after the termination of an activity. Joining is a control that the termination of all preceding activities triggers the start of a succeeding activity.

In the original "Knowledge Convergence Script" design, a four-step process is used. In each phase, three activities are performed in parallel. Only when all activities in one step are finished will all the activities in the next step be triggered. This is a synchronization coordination mechanism. However, if the concurrent tasks performed within a step are not balanced, the efficiency of this coordination mechanism is not high. For example, if one of the three cases is more difficult and takes longer to understand and to develop ideas, then at each step the activity handling this case will take longer. Using synchronization each step takes as long as the most difficult case takes to resolve.

In order to enhance the efficiency a task-driven approach can be used, so that when a learner finishes the current task s/he can perform her/his succeeding activity without having to wait. When there are unbalanced tasks this coordination mechanism can reduce the total learning time. Another possible coordination mechanism is to trigger an activity by an event indicating that all necessary resources are available (data-driven). For example, each learner is responsible for performing four activities: creating an initial report, commenting on two other reports, and writing a final report. Using this approach whenever an initial report written becomes available, the corresponding activity for commenting on it is triggered, even if the learner who will carry it out is still working on her/his initial report.

2) Transfer: In group-based learning, an artifact is usually employed as a means of coordinating group interaction and constitutes a collaboratively produced knowledge object. In the use case, there are transfer dependencies between some activities. For example, artifacts such as initial reports and comments produced in an activity are transferred to other activities.

The basic coordination mechanism for managing transfer dependencies is to capture the artifact produced in the activity and to present the captured artifact in other activities.

3) Usability: In the use case, there are usability dependencies between activities. For example, an initial report of a case should transfer to an activity which has the aim of commenting on this report.

As mentioned above, in e-learning processes, the objects to be transferred are information objects. The coordination mechanisms for managing usability dependencies should check whether the class of the artifacts, data type, size, and other constraints meet the requirements.

Fit dependencies

In the use case each final report is a synthesis of ideas from all the learners, while the production of each report is split into four activities. The use case could be extended so that the three cases are specified as behaviorism, cognitivism, and constructivism, with the three reports being assembled into a general report about learning theories. In this extended case the activities of writing the three reports would have fit dependencies.

A basic coordination mechanism for managing fit dependencies is to check whether the classes of the artifacts, data types, sizes, and other constraints are compatible. A basic coordination mechanism for managing decomposition dependencies between task and sub-task is also needed.

Representation of coordination mechanisms in IMS LD

In this section, we analyze whether IMS LD can represent the coordination mechanisms for managing various dependencies within group-based learning processes which we have identified above.

There are two kinds of activities defined in IMS LD: learning activity and support activity. It is not necessary to distinguish them for our present purpose, and so we simply use the term *activity*. The notations representing resources in IMS LD are *role*, *environment, learning object,* and *learning service*. For the sake of clarity, we discuss these in turn for fit, flow, and sharing.

Representation of Coordination Mechanisms to Manage Fit Dependencies

IMS LD has no notation which explicitly represents artifacts, and so no computational coordination mechanism is available to check whether the components of an artifact fit together. In IMS LD, a general notation *property* can be used to represent a variety of concepts including artifacts created in the learning processes. Depending on its scope, an artifact can be defined as a *global property* or a *local property* (run property). Similarly, an artifact can be defined as a *personal property*, a *role property*, or a *general property*, depending on its owners. A property cannot represent complex, structured information objects because it can only have a primitive data type such as integer, real, string text, URL, file, time, and so on. Consequently IMS LD provides no computational mechanism for coordinating the assembly of components produced simultaneously in different activities. As shown in the use case, the merging work is performed without computational support. Of course, as a general process modeling language, IMS LD should not and cannot directly support any specific artifact. One possible solution is to use a file type suitable for the representation of structured information (e.g., XML files). If external learning services were integrated which checked and assembled components and handled specific artifacts, then the IMS LD engine could communicate with these mechanisms in order to manage the specific fit dependencies. This is a complicated technical issue, however, and so we do not discuss it in detail in this chapter.

Although IMS LD can only manage artifact decomposition dependencies indirectly, it provides several coordination mechanisms which can be used to directly manage task decomposition dependency. In IMS LD a learning process can be decomposed into *plays*, *acts*, and *role-parts*. Each role-part consists of a role and an *activity* or an *activity-structure* that is recursively decomposable. All these notations can be used to represent a set of tasks with a variety of granularities as a hierarchical structure. However, the restriction to activity-structure in which all activities have to be performed by the same role makes it inconvenient to represent a sequence of activities performed, for example, by different roles in turn. If IMS LD had a construct corresponding to a role-part-sequence, then it would be easier to represent a group-interaction sequence involving various roles.

In IMS LD, a role can be decomposed into sub-roles at arbitrary levels. For each role, some attributes can be used to restrict the role, such as max-members, min-members, inclusive/exclusive, and so on. However, no constraint specifies how a role should be composed of sub-roles. As a result it is sometimes difficult to define the formation of a group when it is modeled using role notation. For example, if a group must be formed by three (two female and one male) learners with backgrounds in pedagogy, psychology and computer science respectively, it is difficult to represent such a constraint in IMS LD. As a consequence, no computational mechanism can be used to check whether the group has been correctly formed. We do not go into this in greater detail because there is no a simple method to resolve the issue, and in any event the case under discussion does not raise this particular problem.

In order to provide clarify how to model group interaction in IMS LD without going into too much technical detail, we now introduce a restricted pseudo-code, based on IMS LD. Figure 3 illustrates some definitions of the structure of roles, properties representing artifacts, and activity decompositions. Figure 3a defines three learners: *learner1*, *learner2*, and *learner3*. The constraints for each role are that one and only one user can play a role, and a user cannot have more than one role in this process. The code shown in figure 3b specifies several properties *InitialReport1*, *Comment1-2*, *Comment1-3*, and *FinalReport1*, which represent artifacts produced by *learner1*. Figure 3c defines four activities performed by *learner1*. Each activity will be carried out in an associated environment. Note that the corresponding set of properties and activities relevant to *learner2* and *learner3* are omitted.

```
   <learner create-new="not-allowed" identifier="learner1" match-persons="exclusively-in-roles" max-persons="1" min-persons="1">
      <title>Learner1</title>
   </learner>
   <learner create-new="not-allowed" identifier="learner2" match-persons="exclusively-in-roles" max-persons="1" min-persons="1">
      <title>Learner2</title>
   </learner>
   <learner create-new="not-allowed" identifier="learner3" match-persons="exclusively-in-roles" max-persons="1" min-persons="1">
      <title>Learner3</title>
   </learner>
```

Figure 3a. the definitions of three roles.

```
<!— the definition of the property representing the initial report written by learner1 -->
   <loc-property identifier="InitialReport1">
      <title>Initial Report1</title>
   </loc-property>
<!— the definition of the property representing the comment written by learner1 on the initial
report2 written by learner2 -->
   <loc-property identifier="Comment1-2">
      <title>Comment1-2</title>
   </loc-property>
<!— the definition of the property representing the comment written by learner1 on the initial
report3 written by learner3 -->
   <loc-property identifier="Comment1-3">
      <title>Comment1-3</title>
   </loc-property>
<!— the definition of the property representing the final report written by learner1 -->
   <loc-property identifier="FinalReport1">
      <title>Final Report1</title>
   </loc-property>
```

Figure 3b. the definitions of properties relevant to learner1.

```
<!—the definitions of an activity arranged for learner1 to write a case report -->
<learning-activity identifier="LA-write-initial-report1">
   <title>Write Report1</title>
   <environment-ref ref="ENV-for-report1"/>
   <activity-description>
      <title>Write Report</title>
      <item identifier="ITEM-write-report1" identifierref="RESO-write-report1" />
   </activity-description>
</learning-activity>
<!—the definitions of an activity arranged for learner1 to comment on the InitialReport2
written by learner2 -->
<learning-activity identifier="LA-comment-1-2">
   <title>Learner1 comments on report2</title>
   <environment-ref ref="ENV-for-report2"/>
   <activity-description>
      <title>Commenting</title>
      <item identifier="ITEM-write-comment-1-2" identifierref="RESO-comment-1-2" />
   </activity-description>
</learning-activity>
<!—the definitions of an activity arranged for learner1 to comment on the InitialReport3
written by learner3 -->
<learning-activity identifier="LA-comment-1-3">
   <title>Learner1 comments on report3</title>
   <environment-ref ref="ENV-for-report3"/>
   <activity-description>
      <title>Commenting</title>
      <item identifier="ITEM-write-comment-1-3" identifierref="RESO-comment-1-3" />
   </activity-description>
</learning-activity>
<!—the definitions of an activity arranged for learner1 to write the FinalReport1 -->
<learning-activity identifier="LA-write-final-report1">
   <title>Write Final Report1</title>
   <environment-ref ref="ENV-for-report1"/>
   <activity-description>
      <title>Write Final Report</title>
      <item identifier="ITEM-AD-write-final-report1" identifierref="RESO-write-final-report1" />
   </activity-description>
</learning-activity>
```

Figure 3c. the definitions of activities relevant to learner1.

Figure 3. the definitions of roles, properties, activities.

Representation of Coordination Mechanisms to Manage Flow Dependencies

Precedence: IMS LD provides several built-in mechanisms to manage the precedence dependencies, such as acts in a play and activity-structure with a sequence type. Note that such sequences are weak coordination mechanisms, because the sequences are no more than suggestions. The users can work following the sequences or vary them, because all acts and activities are accessible at any time. They can even access completed activities. This has advantages, because it provides flexibility for the users to carry out tasks as they wish. It is sometimes difficult to judge if an activity has really terminated, especially in learning processes. For example, when learners work on reading and understanding an article and after a period of time they think the task has been finished, they terminate the activity and move on to the next one. However, they may recognize that they did not fully understand the article and go back to read it again. Weak sequence control mechanisms make it possible for users to carry out such tasks flexibly and handle exceptions manually. On the other hand, users have to pay attention to coordination problems, to a greater or lesser extent. Moreover, such freedom for users to decide the actual work sequence may create problems, especially in situations where a strictly defined route is required. Fortunately, IMS LD provides additional mechanisms to support strong controls for sequence of acts and sequence of activities. The following paragraphs will present how weak and strong sequencing mechanisms can be represented in IMS LD.

As shown in Figure 4, the work procedure of this group-based learning is modeled as four acts titled *Case Reporting*, *Criticizing 1*, *Criticizing 1*, and *Finalizing reports*. Each act represents a stage, in which who is responsible for doing which activity is specified as a role-part. In the first act titled *Case Reporting*, for example, *learner1* is assigned to perform the activity titled *Write report1*, which is defined in figure 3 using identifier *LA-write-initial-report1*. Using a weak sequencing mechanism, we can represent four acts in sequence without control as shown in figure 3. However, it is possible to represent a strong sequencing mechanism in IMS LD in a way to specify the completion condition for an act. One such a condition is that an act will be terminated automatically by the system when all role-parts in the act are completed. For example, the first act completes when all learners finish the activities to create their initial reports, and then the activities in the succeeding act titled *Criticizing 1* become accessible.

```
<!—the definitions of four acts in a play -->
<play identifier="PL-work-procedure">
   <title>work-procedure</title>
   <!—the definitions of the first act-->
   <act identifier="ACT-case-reporting">
      <title>Case Reporting</title>
      <role-part identifier="RP-write-report1">
         <title>learner1 writes report1</title>
         <role-ref ref="learner1"/>
         <learning-activity-ref ref="LA-write-initial-report1"/>
      </role-part>
      <role-part identifier="RP-write-report2">
         <title>learner2 writes report2</title>
         <role-ref ref="learner2"/>
         <learning-activity-ref ref="LA-write-initial-report2"/>
      </role-part>
      <role-part identifier="RP-write-report3">
         <title>learner3 writes report3</title>
         <role-ref ref="learner3"/>
```

```
            <learning-activity-ref ref="LA-write-initial-report3"/>
        </role-part>
    </act>
    <!—the definitions of the second act -->
    <act identifier="ACT-criticizing1">
        <title>Criticizing 1</title>
        <role-part identifier="RP-comment-1-3">
            <title>learner1 comments on report3</title>
            <role-ref ref="learner1"/>
            <learning-activity-ref ref="LA-comment-1-3"/>
        </role-part>
        <role-part identifier="RP-comment-2-1">
            <title>learner2 comments on report1</title>
            <role-ref ref="learner2"/>
            <learning-activity-ref ref="LA-comment-2-1"/>
        </role-part>
        <role-part identifier="RP-comment-3-2">
            <title>learner3 comments on report2</title>
            <role-ref ref="learner3"/>
            <learning-activity-ref ref="LA-comment-3-2"/>
        </role-part>
    </act>
    <!—the definitions of the third act -->
    <act identifier="ACT-criticizing2">
        <title>Criticizing 2</title>
        <role-part identifier="RP-comment-1-2">
            <title>learner1 comments on report2</title>
            <role-ref ref="learner1"/>
            <learning-activity-ref ref="LA-comment-1-2"/>
        </role-part>
        <role-part identifier="RP-comment-2-3">
            <title>learner2 comments on report3</title>
            <role-ref ref="learner2"/>
            <learning-activity-ref ref="LA-comment-2-3"/>
        </role-part>
        <role-part identifier="RP-comment-3-1">
            <title>learner3 comments on report1</title>
            <role-ref ref="learner3"/>
            <learning-activity-ref ref="LA-comment-3-1"/>
        </role-part>
    </act>
    <!—the definitions of the final act -->
    <act identifier="ACT-finalizing-report">
        <title>Finalizing reports</title>
        <role-part identifier="RP-write-final-report1">
            <title>learner1 writes the final report</title>
            <role-ref ref="learner1"/>
            <learning-activity-ref ref="LA-write-final-report1"/>
        </role-part>
        <role-part identifier="RP-write-final-report2">
            <title>learner2 writes the final report</title>
            <role-ref ref="learner2"/>
            <learning-activity-ref ref="LA-write-final-report2"/>
        </role-part>
        <role-part identifier="RP-write-final-report3">
            <title>learner3 writes the final report</title>
            <role-ref ref="learner3"/>
            <learning-activity-ref ref="LA-write-final-report3"/>
        </role-part>
    </act>
</play>
```

Figure 4. the definition of a sequence of acts in a play.

In order to support strong precedence dependencies between activities, we can represent the sequence by using conditions to set the visibility of activities. Figure 5 shows an example which supports strong precedence dependency between two activities using a condition. As shown in figure 5, if and only if the first activity, which identifier is *LA-write-initial-report1*, is completed, the second activity, which identifier is *LA-comment-1-3*, becomes accessible. Meanwhile, the first activity becomes inaccessible unless it is specifically set to be visible in other conditions.

```
<if>
   <complete>
      <learning-activity-ref ref="LA-write-initial-report1"/>
   </complete>
</if>
<then>
   <hide>
      <learning-activity-ref ref="LA-write-initial-report1"/>
   </hide>
   <show>
      <learning-activity-ref ref="LA-comment-1-3"/>
   </show>
</then>
<else>
   <hide>
      <learning-activity-ref ref="LA-comment-1-3"/>
   </hide>
</else>
```

Figure 5. the definition of a condition managing a strong precedence dependency between two activities.

The coordination mechanisms discussed above for managing precedence dependencies are task-driven mechanisms. In IMS LD conditions can also be used to represent data-driven mechanisms. For example, if *learner1* submits his/her initial case report, *learner2* can start to perform the activity (its identifier is *LA-comment-2-1*). Otherwise, this activity will be kept hidden from its actor. Figure 6 illustrates this example.

```
<if>
   <not>
      <no-value>
         <property-ref ref="InitialReport1"/>
      </no-value>
   </not>
</if>
<then>
   <show>
      <learning-activity-ref ref="LA-comment-2-1"/>
   </show>
</then>
<else>
   <hide>
      <learning-activity-ref ref="LA-comment-2-1"/>
   </hide>
</else>
```

Figure 6. the definition of a condition representing a data-driven coordination mechanism.

Transfer: IMS LD has no notation which explicitly represents the transference of an artifact produced in an activity and consumed by other activities. However, the transference of an artifact can be represented indirectly. Figure 7 shows an example which transfers an initial report created by *learner1* in the activity *Write Report1* to the activity *Learner2 comments on report1*. Figure 7a shows the definition of the first activity *Write Report1*, in which learner1 writes initial *report1* using the information item *ITEM-write-report1* that refers to a resource *RESO-write-report1*. Figure 7b shows the content of resource file *RESO-write-report1*, in which a global element *set-property* is used to input the initial *report1* captured by the property *InitialReport1*. Figure 7c defines the second activity titled *Learner2 comments on report1* which is associated with the environment *ENV-for-report1*, defined in figure 7d. This environment contains a learning object *LO-information-about-report1*, which has an information item *ITEM-report1*. This item refers to the resource *RESO-presentation-of-report1* and it will become visible when the *InitialReport1* is made available. Figure 7e shows the content of resource file *RESO-presentation-of-report1*, in which a global element *view-property* is used to view the initial report1. In fact the rotation of artifacts is implemented through rotationally binding environments with activities in the original design.

```
<learning-activity identifier="LA-write-initial-report1">
   <title>Write Report1</title>
   <environment-ref ref="ENV-for-report1"/>
   <activity-description>
      <title>Write Report</title>
      <item identifier="ITEM-write-report1" identifierref="RESO-write-report1" />
   </activity-description>
</learning-activity>
```

Figure 7a. The definition of the activity, in which learner1 creates the initial report1.

```
<p>Please write the initial report.</p>
<ld:set-property ref="InitialReport1" property-of="self" />
```

Figure 7b. the content of the resource file "RESO-write-report1".

```
<learning-activity identifier="LA-comment-2-1">
   <title>Learner2 comments on report1</title>
   <environment-ref ref="ENV-for-report1"/>
   <activity-description>
      <title>Commenting</title>
      <item identifier="ITEM-write-comment-2-1" identifierref="RESO-comment-2-1" />
   </activity-description>
</learning-activity>
```

Figure 7c. The definition of the activity that is associated with an environment.

```
<environment identifier="ENV-for-report1">
   <title>working environment for report1</title>
   ……
   <learning-object identifier="LO-information-about-report1">
      ……
      <item identifier="ITEM-report1" identifierref="RESO-presentation-of-report1" isvisible="false">
         <title>report1</title>
      </item>
      ……
   </learning-object>
</environment>
```

Figure 7d. The definition of the environment storing the initial report1.

```
<h3>Initial Report 1:</h3>
```

```
<ld:view-property ref="InitialReport1" view="value"/>
```

Figure 7e. the content of the resource file "RESO-presentation-of-report1".

Figure 7. Transference of an artifact via an environment.

Another solution is to present all imported artifacts in the same information item of the activity which consumes the artifacts. Rather than using an environment, the artifact is transferred by means of the activity-descriptions of the activities which produce and consume the artifact. Because of the limited space available here we omit the code illustrating this approach.

Usability: as mentioned above, in IMS LD a property can be used to represent artifacts. Because a property in IMS LD has a primitive data type such as *integer*, *string*, *duration*, etc. the coordination mechanism for managing usability dependency is simply to check the data type and constraints of the property. In this use case, all properties should be defined as type *text*.

Representation of Coordination Mechanisms to Manage Sharing Dependencies

In IMS LD task allocation is represented as a role-part. As shown in figure 4, a set of role-parts are defined to represent three learners who are assigned to perform different activities. These activities share the labor resources at different times.

We can represent another coordination mechanism for managing sharing dependencies in IMS LD: three sequential activities in each of which three learners work together. Each activity is designed as a collaborative activity leading to the production of a report. Each activity has an environment containing certain learning services such as chat, forum, shared text editor, shared whiteboard, audio/video conferencing, and so on. As mentioned before, in a fluid collaboration learners can use these collaborative tools to coordinate their actions at a finer-grained level and produce shared artifacts. Because the code representing this coordination mechanism is extensive it is not included here.

IMS LD provides static coordination mechanisms for managing sharing dependencies, but it is difficult to support dynamic coordination mechanisms, for example, the "first come, first served" mechanism. We can investigate how to model an alternative design, in which tasks are assigned to roles according to the time sequence that users register to the execution. Using this approach it is unpredictable at design time who will come first in an actual execution, unlike a pre-defined allocation of tasks as role-parts described in figure 4. Because the XML code to implement this mechanism is too extensive we describe and explain it using pseudo-code as shown in figure 8.

In order to control the execution of activities at the right time, data-driven mechanisms (similar to the code shown in figure 6) are needed as a complete coordination mechanism. Figure 8a declares three roles: *learner1*, *learner2*, and *learner3* and fifteen activities: three registering and twelve activities illustrated in figure 1. Figure 8b declares three activity-structures and each activity-structure consists of four sequential activities: writing the initial report, commenting on the reports of two peers, and creating the final

report. Figure 8c defines three properties representing the time when learners finish the registration. Figure 8d specifies how the values of three properties are assigned. Because three learners may complete registration at different points of time, the *current time* assigned by the system will have different values for different learners. In figure 8e, the first statement specifies that if *learner1* and *learner2* have registered and *learner3* has not finished registration, and *learner1* registered before *learner2* did (or they registered at the same time) then *learner1* will be assigned to perform *activity-structure1*, *learner2* will be responsible for doing *activity-structure2*, and *activity-structure3* will be carried out by *learner3*. The following five statements specify the allocation tasks in the other five situations, in which three learners finish the registrations in different time sequences.

```
<!— the three roles and twelve activities are defined as those defined in the original design. -->
Role: learner1, learner2, learner3;
Activity: registering1, registering2, registering3, reporting1, ....., revising3;
```

Figure 8a. the declaration of three roles and twelve activities.

```
<!— the four activities performed by the same learner are defined as a sequence activity-
structure. Therefore, three activity-structures are defined -->
Activity-structure: activity-structure1 := reporting1 + criticizing1-3 + criticizing1-2 + revising1;
                    activity-structure2 := reporting2 + criticizing2-1 + criticizing2-3 + revising2;
                    activity-structure3 := reporting3 + criticizing3-2 + criticizing3-1 + revising3;
```

Figure 8b. three properties are defined for representing when each learner registers.

```
<!— three properties are defined for representing when each learner registers -->
Property: T1 := 0, T2 := 0, T3 := 0;
```

Figure 8c. when a learner has finished registration, the registration time will be recorded.

```
<!— when a learner has finished registration, the registration time will be recorded -->
If (registering1 complete) then T1 := current time;
If (registering2 complete) then T2 := current time;
If (registering3 complete) then T3 := current time;
```

Figure 8d. according to the sequence in which three learners register, the activity structures will be assign to the learners in the way first-come-first-served.

```
<!— according to the sequence in which three learners register, the activity structures will be
assign to the learners in the way first-come-first-served -->
If ((T1 is not 0) and (T2 is not 0) and (T3 is 0) and (T1<=T2)) then notification (learner1 activity-
structure1), notification (learner2 activity-structure2), notification (learner3 activity-structure3);

If ((T1 is not 0) and (T2 is not 0) and (T3 is 0) and (T1>T2)) then notification (learner1 activity-
structure2), notification (learner2 activity-structure1), notification (learner3 activity-structure3);

If ((T1 is not 0) and (T2 is 0) and (T3 is not 0) and (T1<=T3)) then notification (learner1 activity-
structure1), notification (learner2 activity-structure3), notification (learner3 activity-structure2);

If ((T1 is not 0) and (T2 is 0) and (T3 is not 0) and (T1>T3)) then notification (learner1 activity-
structure2), notification (learner2 activity-structure3), notification (learner3 activity-structure1);

If ((T1 is 0) and (T2 is not 0) and (T3 is not 0) and (T2<=T3)) then notification (learner1 activity-
structure3), notification (learner2 activity-structure1), notification (learner3 activity-structure2);

If ((T1 is 0) and (T2 is not 0) and (T3 is not 0) and (T2>T3)) then notification (learner1 activity-
structure3), notification (learner2 activity-structure2), notification (learner3 activity-structure1);
```

Figure 8e. notifications are used to allocate tasks dynamically.

Figure 8. an example of dynamic coordination mechanism.

If notification is not used, it is necessary to enumerate all possible role-parts in the same act (the total number of turples is the combination of the number of roles and the number of activities, 3*12=36 in this use case), and set them to *invisible*. After the rotation pattern is determined, 12 activities are set to *visible* to make 12 associated role-parts active. If the number of users and cases increases, the complexities of the process model increase accordingly. The difficulties in representing dynamic coordination mechanisms are ascribed to a) no *identifier* data type and no *collection* data type specified for the property and b) insufficient operations such as 'find a person whose personal property meets a condition', 'add a person as an active role', 'add a role-part within an act', and so on.

FUTURE TRENDS: THE REUSE OF COORDINATION MECHANISMS

As we have seen, representing coordination mechanisms is a time-consuming and error-prone task. It is necessary to explore whether coordination mechanisms can be represented at a more abstract level than XML, that is to say at a higher level than the executable code. It is expected that the abstract representation could be more intuitively understood and used by practitioners (e.g., instruction designers and teachers) who do not have sophisticated technical knowledge and skills. The system would then automatically transform such an abstract representation into XML code. This process provides a means whereby coordination mechanisms could be reused without requiring users to understand how the executable code works. In this section we discuss issues related to such reuse.

Identifying common dependencies and the mechanisms for managing them

According to coordination theory, dependencies and the mechanisms for managing them are *general*, which means that a given dependency and a mechanism to manage it will be found in a variety of settings. For example, a common coordination problem appears when certain activities require specialized competences, thus constraining which persons can work on them. This kind of dependency arises in many situations and there is a generic set of coordination mechanisms (managing this dependency) which appear over and over in different processes. Coordination theory also describes how several coordination mechanisms can often be used to manage a dependency. For example, mechanisms to manage sharing a dependency between roles and activities can include qualification-checking, priority-comparing, first-come-first-served, and so on. Because of this it is valuable to identify and study common dependencies and their related coordination mechanisms, in order to facilitate reuse.

Reusing computational coordination mechanisms

Once the dependencies and corresponding coordination mechanisms have been identified, the next step is to represent the coordination mechanisms in IMS LD. As we have seen, the representation of some coordination mechanisms in IMS LD is a very complex task, even for users with sound technical knowledge. It is therefore desirable to make the representation of coordination mechanisms reusable. Through an analysis of the IMS LD manifest file and resource files, we have found that some parts of code are static and some parts of code are replaceable and related to particular elements. We can therefore store a fragment of code as an executable component in a library of an IMS LD authoring environment. We can refer to this using an abstract representation, which can have parameters with values which are assigned by the user in design-time. For example, if a user wants to model the transference of a document from one activity to another, s/he can use an abstract representation: *transfer a document (parameter1) from an activity (parameter2) to another activity (parameter3)*. The constraints for the parameters are that *parameter1* must be a property reference representing a document to be transferred, *parameter2* and *parameter3* must be activity references. Once the user has applied a coordination mechanism (by choosing the corresponding abstract representation and assigning the values to parameters) the system automatically maps the abstract representation to the component.

In the same way, more complex coordination mechanisms needed in the 'Knowledge Convergence Script' can be represented as well. For instance, the abstract representation: *distribute documents (document1, document2, document3) within activities (activity1, activity2, activity3)* indicates the one-to-one distribution of three documents between three activities. Similarly the abstract representation: *rotate documents (document1, document2, document3) from activities (activity1, activity2, activity3) to succeeding activities (activity4, activity5, activity6)* means to transfer three documents produced in three activities to three succeeding activities as follows:

- transferring document1 produced in activity1 to activity5
- transferring document2 produced in activity2 to activity6
- transferring document3 produced in activity3 to activity4.

It is clear that a high-level representation of coordination mechanisms of this kind is much easier to understand and use than a concrete representation codified using IMS LD and expressed in XML (see figure 5), or using a programming language (e.g., JAVA). Currently, we are working on developing a high-level modeling language and mapping algorithms to transform a group-based learning design represented in the high-level modeling language to an executable model represented in IMS LD. This work is technical in nature, and so we do not discuss the details in this chapter.

CONCLUSIONS

This research is a theory-based analysis. First, we briefly introduce group-based learning and coordination theory. Using coordination theory as an analytical framework we

analyze dependencies and possible coordination mechanisms for managing them in group-based learning. We identify a variety of dependencies and some related coordination mechanisms through the investigation of a use case and some of its variants. We then analyze the expressiveness of IMS LD in representing the identified coordination mechanisms. We conclude that in supporting group interaction it is possible to represent almost all basic coordination mechanisms in IMS LD. In particular, IMS LD provides sufficient mechanisms to manage: task and role decomposition dependencies, weak and strong precedence dependencies, and static resource sharing dependencies. However, we have also recognized that the representation of certain coordination mechanisms presents some challenges. Specifically it is complex to represent: the coordination of the assembly of components, transference of artifacts in some complicated distribution patterns, complicated group formation and group dynamics, and allocation of tasks and resources using some dynamic coordination mechanisms. The reasons for these difficulties are briefly analyzed and possible solutions are also discussed.

Based on this analysis, we have briefly explored the feasibility of reusing coordination mechanisms in modeling group-based learning processes. In comparison with IMS LD code in the form of XML, a representation of common coordination mechanisms at a high-level of abstraction may be more intuitively understood and used by practitioners. We are currently identifying and codifying generic coordination mechanisms which will be archived as a library in the IMS LD authoring environment for reuse on future occasions. We will implement an advanced IMS LD authoring environment in which the user can design group-based learning processes using the abstract representation. The system will then automatically generate IMS LD code based on abstract representations and the executable components in the library.

ACKNOWLEDGEMENT

The work on this paper has been sponsored by the TENCompetence Integrated Project that is funded by the European Commission's 6th Framework Programme, priority IST/Technology Enhanced Learning. Contract 027087

REFERENCES

Caeiro, M., Anido, L. & Llamas, M. (2003). A Critical Analysis of IMS Learning Design. *In Proceedings of CSCL 2003*, p.363-367.

Crowston, K. and Osborn, C. (1998). *A coordination theory approach to process description and redesign*, Massachusetts Institute of Technology, Centre for Coordination Science.

Dillenbourg P. (1999) What do you mean by collaborative learning? In P. Dillenbourg (Ed) *Collaborative-learning: Cognitive and Computational Approaches* (pp.1-19). Oxford: Elsevier.

Dillenbourg, P. (2002). Over-scripting CSCL: The risks of blending collaborative learning with instructional design. In P. A. Kirschner (Ed). *Three worlds of CSCL. Can we support CSCL* (p. 61-91). Heerlen, Open Universiteit Nederland.

Fischer, F., Kollar, I., Mandl, H, and Haake, J. (2007). *Scripting Computer-Supported Collaborative Learning: Cognitive, Computational and Educational Perspectives*. Springer.

Hernandez, D., Asensio, J.I., Dimitriadis, Y. (2004). IMS Learning Design Support for the Formalization of Collaborative Learning Flow Patterns. Proceedings of the 4th *International Conference on Advanced Learning Technologies*, pp.350-354, Aug.30 - Sep. 1, 2004, Joensuu, Finland, IEEE Press.

IMSLD. (2003). *"IMS Learning Design Specification."* Retrieved September 15th, 2006, from http://www.imsglobal.org/learningdesign/index.cfm.

Johnson, D.W (1981). Student-student interaction: the neglected variable in education. *Educational Research*, 10, 5-10.

KCS uol (2007). Retrieved July 22, 2007, from http://hdl.handle.net/1820/1003.

Kollar, I., Fischer, F., & Slotta, J.D. (2005). Internal and external collaboration scripts in web-based science learning at schools. *In: Proceedings of Computer Supported Collaborative Learning (CSCL2005): The Next 10 Years*, 331-340, Mahwah, NJ: Lawrence Erlbaum Associates.

Koper, R. and Olivier, B. (2004). "Representing the Learning Design of Units of Learning." *Journal of Educational Technology & Society* 7(3): 97-111.

Malone, T. W. and Crowston, K. (1994). "The interdisciplinary study of coordination." *ACM Computing Surveys* 26(1): 87-119.

Miao, Y., Hoeksema, K., Hoppe, U., & Harrer, A. (2005). CSCL Scripts: Modelling features and potential use. *in: Proceedings of Computer Supported Collaborative Learning (CSCL2005): The Next 10 Years*, 423-432, Mahwah, NJ: Lawrence Erlbaum Associates.

O'Donnell, A. M., & Dansereau, D. F. (1992) Scripted Cooperation in Student Dyad: A Method for Analyzing and Enhancing Academic Learning and Performance. In R. Hertz-Lazarowitz and N. Miller (Eds.), *Interaction in Cooperative Groups: The theoretical Anatomy of Group Learning* (pp. 120-141). London: Cambridge University Press.

Strijbos, J.W. & Martens, R.L. (2001) Group-based learning: Dynamic interaction in groups. *Paper presented in EURO-CSCL Conference* 2001, March 22-24, Maastricht, The Netherlands.

Strijbos, J.W., Martens, R.L., & Jochems, W.M.G. (2004) Designing for interaction: Six steps to designing computer-supported group-based learning. *Computer & Education* 42, 403-424.

Tribe, D.M.R., (1994). An Overview from Higher Education, in *Using Group-based Learning in Higher Education*, L. Thornley and R. Gregory, Editors. Kogan Page Limited: London. p. 25 - 31.

Van Es, R., & Koper, R. (2006) Testing the pedagogical expressiveness of IMS LD. *Educational Technology & Society*, 9 (1), 229-249.

Vogten, H., Martens, H., Tattersall, C., Van Rosmalen, P, Nadolski, R, and Koper, R. (2006). CopperCore Service Integration - Integrating IMS Learning Design and IMS Question and Test Interoperability. *In Proceedings of The 6th IEEE International Conference on Advanced Learning Technologies*, 378-382, Kerkrade, The Netherlands, IEEE Computer Society.

Weinberger, A., Fischer, F., and Mandl, H. (2004). Knowledge convergence in computer-mediated learning environments: Effects of collaboration scripts. *Proceedings of the 85th Annual Meeting of the American Educational Research Association (AERA),* San Diego, CA, USA.

Weinberger, A., Stegmann, K., and Fischer, F. (2005). Computer-supported collaborative learning in higher education: Scripts for argumentative knowledge construction in distributed groups. *In: Proceedings of Computer Supported Collaborative Learning (CSCL2005): The Next 10 Years*!, 717-726, Mahwah, NJ: Lawrence Erlbaum Associates.

Definitions of Key Words

Group-based learning is an instructional strategy in which a small group of learners work together in a series of activities in order to achieve a shared learning objective.

Coordination is the process of managing dependencies between activities (Malone and Crowston, 1994).

A *coordination mechanism* refers to additional activities that can be used to manage dependencies (Malone and Crowston 1994).

IMS LD is an open e-learning technical standard used to model teaching and learning processes.

Learning design is a description of a series of activities aiming at achieving learning objectives. In this chapter the term *learning design* normally refers to the description of the learning process in IMS LD

CSCL script is a formal description of an online collaborative learning design.