Architecture and conceptual model of Virtual Campus

The virtual campus team

11th June 2003

1 Introduction

The Polimi Virtual Campus platform is an open system for the design, deployment, fruition, and evaluation of reusable learning materials. Its main objectives are:

• To support design, composition, and reuse of Learning Objects (LOs).

• To support fruition of Learning Objects in individual and cooperative usage.

• To support analysis of students’ behavior (learning, relational, and normative aspects).

• To analyze power consumption in all envisaged scenario.

• To experiment the usage of our learning platform and of advanced learning objects in our CS courses.

This report provides a short introduction to the architecture and conceptual model of the Virtual Campus platform. More general information on the platform are provided at the URL http://www.elet.polimi.it/res/vcampus.

2 High level architecture of Virtual Campus

The Virtual Campus platform is composed of two main subsystems (see Figure 1).

The Authoring Environment supports Educators in creating and organizing didactical material in Learning Objects (LOs). The internal structure of the authoring environment is described in more detail in Section 3.2, and exploits proper languages for LO definition that are described in Sections from 3.3 to 3.6.

Learning Objects produced within the authoring environment are then serialized through the SCORM & XLANG generators, so that they are available to the Fruition
Figure 1: Virtual Campus high level architecture.
environment. It enables fruition of LOs by Learners. Such a fruition can be guided by Educators, and can involve either individuals or groups of Learners. The core of the fruition environment is based on a process engine that enacts the fruition workflow associated to a LO, guiding Learners and Educators in the execution of the activities related to the usage of the LO. The fruition environment includes tools used to exploit all contents associated to LOs. These tools other than standard productivity tools such as Microsoft Powerpoint or Acrobat Reader and tools supporting synchronous collaboration such as Microsoft NetMeeting, include tools that we have specifically developed within the Virtual Campus project. They are:

- **Lezi** that allows the users to visualize various kinds of related contents (e.g., audio/video sequences and slides), in a synchronized way.

- **PeerVersy** that supports configuration management in a peer to peer setting and is used by students sharing a project who are able to work autonomously and synchronize their work when possible.

- **WebTalk** that provides an interesting 3-D metaphore to browse through didactical material. The tool allows the user to maintain awareness of presence of other users browsing through the same material thus enabling less structured interaction and sharing of information among learners.

- **WebBoardExport** that, starting from the structure of a LO (typically, a CLO corresponding to a whole course or to a course module), configures automatically a virtual discussion environment on specific topics associated to the LO. The discussion engine is based on the WebBoard Conferencing Server commercial product that allows users to collaborate using message boards, chatting, and exchanging documents.

The Virtual Campus platform is instrumented with a monitoring and evaluation environment that collects data on actions performed by users. Such information is stored in profiles that are associated to both LOs and Learners, and is used, on the one side, to provide feedbacks to Educators on the validity of specific learning objects and on the behavior of Learners, and, on the other side, to instruct automated tutoring agents that give suggestions to Learners and guide them during the fruition process. In particular, tutoring agents correct wrong learning styles. As an example, learners can tend to isolate themselves, choosing stand-alone LOs instead of collaborative ones. Observing the behavior of learners, agents are able to suggest them to use more collaborative LOs in order to increase their workgroup capabilities.

### 3 The Virtual Campus conceptual model

#### 3.1 Learning Objects

The core concept in the Virtual Campus conceptual model is *Learning Object* (LO). A learning object is anything that can be used to vehiculate some concept from a Teacher
to a Learner. For instance, a lesson, an exam, but also an entire course are all modeled as LOs. Thus, LOs can either be elementary items that we call *Atomic Learning Objects* (ALOs) or they can be defined as the aggregation of other LOs. These last ones are called *Complex Learning Objects* (CLOs). While ALOs may represent single lessons or learning units and have associated specific contents such as slides, videos, tests, homeworks, etc., CLOs represent courses or part of courses. Being composed of various LOs (either ALO or CLO), they establish some relationships between component LOs (see Section 3.4 for more details). At fruition time, relationships may force the order in which Learners exploit such LOs. They are defined by Organizers and can be refined and customized by Teachers before making them available for fruition.

Fig. 2 shows the main concepts we have introduced so far and the way they are related.

![Diagram](image)

Figure 2: Learning Objects and their relationships.

### 3.2 Authoring activities and support offered by the Authoring environment

In order to enable separation of roles and of concerns, the authoring process leading to the development of LOs is structured in the following activities:

**Definition of contents**  this activity is devoted to the production of content material such as slides, videos and others. The VC platform does not explicitly support such activity but relies for this purpose on the existence of various well established tools (examples are document editors such as Microsoft Office or LaTeX).
Definition of Reusable ALOs Contents built in the previous activity are associated to proper metadata with the purpose of defining items that can be stored in a repository, searched, and reused in the design of courses. Metadata aim at providing synthetic information on ALOs that helps the users of the VC platform in understanding their purpose, knowing their originator, etc. Metadata we have adopted are mainly those of the IEEE LOM, but we have introduced some extensions. For instance we allow the definition of various preconditions that constraints the fruition of LOs. Such extensions are presented in detail in Section 3.3. Reusable ALOs are denoted as ALOsR.

Definition of Reusable CLOs As ALOsR, reusable CLOs have associated proper metadata describing their general properties. In addition, they define a graph describing all relationships between component LOs. An example is given in Fig. 3.a where the CLO Calculus is shown. Component LOs are represented either as nodes or as boxes containing other nodes. The arrows between the elements of the graph are relationships. For instance, the relationship IsRequiredBy between Functions and Limits indicates that before taking the Limits LO, the Functions LO has to be completed. More details on the graphical language for defining Reusable CLOs and on its semantics are given in Section 3.4.

Definition of Didactical-level CLOs Didactical-level CLOs represent specializations of Reusable CLOs that depend on the specific course in which they are used and on the specific Teacher. For instance, given the LO describing a Foundation of Computer Science course, the Teacher might need to make it specific for a mechanical engineering degree. This could be done by eliminating some in depth studying on dynamic data structures and complexity theory and by forcing the lesson on matrices to be given as soon as possible so that students can then use the resulting knowledge within the parallel Mathematics course. To support this customization activity, the workflow guiding fruition is derived from the Reusable CLO. It can then be specialized by Teachers and further constrained, compatibly with the relationships defined in the Reusable CLO. Fig. 4.a shows an example of Didactical-level CLO derived from the Calculus CLO shown in Fig. 3.a. The transformation is partially automated and it is presented in detail in Section 3.5. In order to distinguish between Reusable and Didactical-level CLOs, we denote the first ones as CLOsR and the second ones as CLOsD.

Definition of Fruition-level LOs Before making LOs (either ALOs or CLOsD available for fruition, other details need to be added concerning the specific edition of a course or a lesson where they are used. Examples are the names of students who will attend the course, the course calendar, etc. This activity leads to the definition of LOsF (ALOsF and CLOsF) that are ready to be installed in the fruition environment. In order to make LOsF exportable also to other e-learning environments, they are serialized according to an extension of the SCORM format. The extension takes into account the metadata introduced in the VC-LOM and is simply disregarded by non VC-compliant tools. In addition, the workflow associated to CLOsF is serialized in XLANG [], the XML-based
process language that is interpreted by the process engine we have adopted (BizTalk).

Summing up, all authoring activities are performed at three main levels of abstraction: the reusable-level where general-purpose LOs\(_R\) are defined; the didactical-level where the CLOs\(_D\) didactical workflow is defined starting from the constraints defined at the higher level; the fruition level where additional details are added, thus leading to the definition of CLOs\(_F\) and ALOs\(_F\).

Consistently to the authoring activities and the levels listed above, the Virtual Campus authoring environment provides editors to define ALOs\(_R\) and CLOs\(_R\), a CLO\(_D\) generator and editor that automatically generates a first version of the workflow associated to a CLO and then supports Teachers in customizing it, and, finally, a LO\(_F\) tailoring tool that supports the insertion of all fruition-level details.

The next sections present the languages used in the various authoring phases.

### 3.3 The ALO\(_R\) language: VC LOM

Metadata for VC are specified according to an extension of IEEE LOM that we have called VC LOM. The extensions mainly concern metadata of the Educational category. The most important items we have added are:

Metadata are described according to an extension of IEEE LOM that we have called VC LOM. The extensions mainly concern metadata of the Educational category. The most important are:

- **Expiration Time**: the time when the learning object has to be considered expired and therefore can be eliminated from the repository.

- **Synchronism Attribute**: it specifies the type of time dependency of the LO execution, i.e. if the LO must be executed synchronously (all learners must start fruition at the same prearranged execution time) or may be executed asynchronously.

- **Cooperation Attribute**: it specifies if the learners execute the LO in cooperation with other learners or not.

- **Auditing Attribute**: it specifies if during the execution of a LO external observers are allowed or not. External observers can be other learners, tutors, evaluators, and observe in a passive way the process of consumption of LOs’ contents.

- **Supervision Mode**: it defines the level of supervision of learners activities. Predefined levels are “none” (no supervision), “tutored” (learners can ask the supervisor attendance), “supervised” (the supervisor is always present during the studying process), “driven” (learners must study in a passive way, strictly following the supervisor’s instructions).

- **Group Cardinality**: it defines the cardinality of the group involved into the fruition of the LO. Meaningful group cardinalities are “1” (self studying), “2” (pair studying, both learner-learner and learner-instructor), “m” (group studying).
• Preconditions Time Constraints: the fruition of LOs can be constrained to one or more time instants, e.g., the start time.

• Preconditions User Profile Properties: the administrative background and profile a learner should have before exploiting the LO.

• Preconditions Educational Requirement Specification: the skills and knowledge a learner should have in order to exploit the LO successfully. It is defined in general as a predicate on learning objectives of other LOs (eventually external to the virtual campus system).

• Learning Objectives Time Constraints: the fruition of LOs can be constrained to one or more time instants, e.g., the end time.

• Learning Objectives User Profile Properties: the administrative objectives that a learner reaches by exploiting the LO.

• Learning Objectives Educational Requirement Specification: specification of educational objectives of the LO in terms of skills and knowledge improvement a learner can obtain exploiting it.

By using the VC LOM metadata extension, we may express explicitly some important properties of the fruition process of LOs. So, we can say that LOs may be exploited synchronously or asynchronously, in cooperative or not cooperative way, with or without observers, with or without supervisors, relied to a pre-arranged group cardinality where both learners and instructors are involved. Also, other meta attributes allow to specify the relationships between a LO and the student profile required for the fruition, and the skills obtained after a LO fruition. Such attributes specify preconditions and learning objectives (both educational and temporal), student profile properties that must be true before the execution of LOs, and define as the fruition of LOs can modify (improve, in general) a user profile. The other metadata of the VC LOM are the same of IEEE LOM. Interested readers may refer to IEEE\textsuperscript{1} for a complete and detailed description.

3.4 The CLO\textsubscript{R} language

As shown in Fig. 3, the CLO\textsubscript{R} language is graphical, but it has a corresponding serialization in the VC LOM. For space reasons we do not describe the details of such serialization that concerns trivial syntactic transformations.

In the CLO\textsubscript{R} graph, nodes univocally represent LOs (either ALOs or CLOs). Test-LOs are decorated by a T on the right handside of the corresponding node.

Rounded-corners rectangles drawn inside a CLO are Inner-CLOs. They provide a mechanism to aggregate LOs, but, differently from other LOs, they do not have an identity and cannot be reused outside the context of the CLO in which they are defined.

\textsuperscript{1}IEEE: Learning Technology Standardization Committee. Standard for Learning Object Metadata. \url{http://ltsc.ieee.org}
Relationships indicate the presence of didactical constraints that link two LOs in the context of the containing CLO. A generic relationship from $x$ to $y$ in the context of $z$, with $x, y$ being nodes (either ALOs or CLOs) and $z$ a CLO, is represented by an arrow that starts from $x$ and ends to $y$, inside $z$; the arrow is labeled with the relationship name. Supported relationships are the following:

- **IsRequiredBy** $Node_A IsRequiredBy Node_B$ relationship indicates that $Node_A$ must be completed before entering $Node_B$. In other terms, the Learner has to possess $Node_A$-related knowledge in order to achieve a correct understanding of $Node_B$ content. Satisfying the IsRequiredBy constraint is a necessary (but not sufficient) pre-condition for entering $Node_B$. However, the IsRequiredBy relationship does not mean that Learners must complete $Node_A$ immediately before $Node_B$: Learners are allowed to make use of other Nodes after $Node_A$ and before $Node_B$ fruition. The presence of several IsRequiredBy entering the same Node is equivalent to an AND boolean operator among the relationships. IsRequiredBy is antisymmetric, non-reflexive and transitive.

- **IsAlternativeTo** $Node_A IsAlternativeTo Node_B$ relationship indicates that $Node_A$ and $Node_B$ are mutually exclusive, although both valid since their didactical function is considered to be identical. Two nodes connected by an IsAlternativeTo relationship are automatically enclosed within an inner-CLO. Consistently, all relationships connecting these two nodes with the other nodes in the graph end to or start from the box delimiting the inner-CLO. Moreover, none of the two nodes can be singularly involved in other relationships. This is to ensure that the...
two nodes can be actually used alternatively without implying any change in the consequent learning paths. For the sake of simplicity, no more than one IsAlternativeTo can enter or exit a single Node. IsAlternativeTo is symmetric, non reflexive and transitive. In Fig. 3.a the two inner CLOs on the left are connected by an IsAlternativeTo relationship.

- **References** \( \text{Node}_A \) References \( \text{Node}_B \) relationship indicates that \( \text{Node}_A \) cites \( \text{Node}_B \) for more details on some topic related to \( \text{Node}_A \) itself. Taking \( \text{Node}_B \) at fruition time is not compulsory: Learners can thus decide whether to make use or to ignore this information. Many References can enter or dipart from the same Node. In this last case, Learners can make use of one or more of the corresponding LOs. Only leaves in the graph representing a CLO can be referenced (i.e., they can be the termination node of a References relationship). Indeed, referenced Nodes cannot be the termination point of types of relationships other than References. References is antisymmetric and non-reflexive.

- **RequiresOnFailure** A RequiresOnFailure relationship always connects a Test-LO with some other Node. If the Test-LO is failed, then the Node at the other end of the RequiresOnFailure relationship has to be taken by the learner. Two cases are possible:
  - The terminating Node of the RequiresOnFailure is a leaf of the learning path. In this case, it is intended that as soon as it is completed, the Learners goes back to take the Test-LO again.
  - The terminating Node of the RequiresOnFailure is part of the path that has been already visited by the Learner before taking the Test-LO. In this case that fragment of CLO starting from that Node and leading to the Test-LO is taken again as it was the first time. Any intermediate achievement that has been previously obtained by the Learner is disregarded.

If no RequiresOnFailure is specified, learners have to re-start the fruition of the whole CLO. RequiresOnFailure is antisymmetric and non- reflexive.

Fig. 3.a shows definition of CLO “Calculus”. It is reused into CLO “Mathematics” (Fig. 3.b), that, in turn, belongs to the “Engineering, first year” CLO (Fig. 3.c). The CLO “Calculus” is composed by an Inner-CLO and the LO “Derivatives”. Learners have to make use of LOs inside the Inner-CLO before studying “Derivatives”. The Inner-CLO, in turn, is composed by four LOs; ALOs “Functions” and “Limits” represent a branch and are alternative to the branch composed by LOs “Introduction” and “Functions & Limits”. Learners are allowed to make use of the former branch or the latter but not of both.

Inside CLO “Mathematics” (Fig. 3.b), after the completion of the LO “Basic concepts”, learners may decide to take the LO “History of mathematics” or not. In any case, learners are then allowed to make use of the Inner-CLO holding “Calculus” and “Geometry” (learners have to study both LOs in any order) and finally they can try
the LO “Test”. If learners fail the test, they have to re-start the fruition from the Inner-CLO.

When the “Calculus” CLO is reused within “Mathematics”, it is seen as any other LO. That is, its internal structure is completely hidden. This enables separation of concerns and facilitates compositionality of LOs.

3.5 The CLO\textsubscript{D} language

At the Didactical Level, the system converts LOs into a “workflow-like” representation using UML Activity Diagrams [ ]. Then, the system gives Teachers the opportunity to customize resulting CLOs\textsubscript{D}. Fig. 4 shows the workflows generated starting from the CLOs\textsubscript{R} shown in Fig. 3. The two notations are quite similar. Nodes in a CLO\textsubscript{R} correspond to activities in the CLO\textsubscript{D} workflow. Such activities can either be atomic or composite. Atomic activities correspond to ALOs. Their activation at fruition time consists in sending some message to the user (e.g., please click on this URL) or activating some tool (e.g., Powerpoint or any other tool used to exploit some content). Composite activities correspond to CLOs. At fruition time their activation consists in instantiating the corresponding workflow. Two activities connected by a direct arrow as Matematics and Physics in Fig. 4.c are executed in a sequence. Such a sequence can be interleaved if other parallel activities exist. This is the case of Chemistry that can be executed before Matematics, after Physics, or between the two. In UML activity diagrams the two vertical bars of Fig. 4.c enclose parallel paths that are executed independently. The label <<Optional>> in Fig. 4.b represent a so called stereotype that we have introduced to personalize the semantics of the arrow connecting History of mathematics to the diamond. At fruition time such an arrow is followed in the two directions in all cases the Learner taking the CLO decides to activate History of mathematics. Otherwise it is simply disregarded. In Fig. 3.a, the paths Functions & Limits and the one Introduction and Funct.+Lim. are alternative. Any of them has to be taken before taking the Derivatives LO.
the corresponding CLO\textsubscript{R} description. The Teacher can decide to customize them by performing one of the following operations:

- Elimination of alternative paths.
- Elimination of optional activities.
- Forcing fruition order in case of parallel activities.

All aforementioned operations preserve the consistency of the resulting workflows and further constrain the semantics conveyed by the Organizer who has defined the corresponding CLO\textsubscript{R} thus limiting the set of didactical opportunities that are given to Learners. This might be convenient in all cases where Learners do not have enough experience and maturity to make their own choices. Other operations could also be envisaged, for instance, to enable elimination of intermediate elements in the workflow or addition of new parts. At this stage we do not have a definitive answer on the validity of such possibilities. We plan to evaluate these and the ones we have already implemented based on the feedbacks that we will gather from the use of the Virtual Campus environment.

3.6 The LO\textsubscript{F} metadata and the serialization to SCORM

At fruition level the most important entity is the course. In general, a course is related to a CLOD. It has some metadata such as title, description, subject area, difficulty, version, enroll method, start and end date, and other associated data like course calendar, announcements, lectures, teacher, tutors, learners. All this information falls into the fruition level of VC. LO\textsubscript{F}s are less reusable with respect to LO\textsubscript{R}s and LO\textsubscript{D}s. In fact, the reusability of LOs shifts while we move from the reusable level to the fruition level. This is also an additional reason to have different levels of LOs’ definition in the authoring environment: to provide several levels of reuse of didactical resources. LO\textsubscript{F}s are serialized following the SCORM packaging format when they are exported to the fruition environment. Roughly speaking, a course is exported in a zip-compressed archive (according to the SCORM packaging recommendation) that contains all course data (raw contents) and metadata. In particular, the zip archive is composed by a XML manifest, which contains the references to metadata of LO composing the course and the corresponding structure, a metadata section, which contains a different XML file holding the metadata for each LO belonging to the course, a course fruition section, that store in XML format some typical fruition data as the course calendar and provided announcements, an export section which contains a log of the export process of the course, some JavaScript API that provide functions to call the runtime of the fruition environment (according to the SCORM Run Time Environment recommendation), and finally the LOs’ raw contents and users’ data (teacher, tutors, learners). To be more precise, the VC serialization algorithm of a LO\textsubscript{F} introduces an extension of the SCORM packaging format. In fact, there is support for the whole VC LOM, including the item added to the IEEE LOM (which instead is used in the SCORM packaging recommendation to define serializable
metadata). Furthermore the workflow related to the CLOFs associated to the exported course is serialized in a XLANG schedule. Such an XLANG schedule can be replaced by a SCORM workflow description as soon as the corresponding workflow language will be released.