We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,300

Open access books available

172,000

International authors and editors

190M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Educational Modelling Languages and Service-Oriented Learning Process Engines

Jorge Torres¹, César Cárdenas¹, Juan Manuel Dodero² and Eduardo Juárez¹

¹*The Distributed and Adaptive Systems Lab for Learning Technologies Development
Tecnológico de Monterrey
Mexico*

²*Computer Science Department
Universidad de Cádiz
Spain*

1. Introduction

E-Learning Systems evolution (Torres et al., 2005) has offered an enhanced set of characteristics and flexibility to the learning process, starting from simple static Web contents, moving to the standardization of learning objects, and giving place to the creation of Educational Modelling Languages (EML) able to fully describe learning scenarios.

EMLs (Koper, 2001) represent an important approximation to integrate diverse educative aspects, allowing the design and implementation of spaces and activities supported by Information and Communication Technologies. With EMLs it is possible to integrate, under a learning process, personalized educative material for each student –e.g. activities, services, resources, objectives, evaluations, profiles–, promoting the active participation of the student with his learning process.

As EMLs evolve, design issues and more needs have been identified giving place to new desirable EML characteristics, such as pedagogical diversity, learning flow description, dynamic and unanticipated composition, separation of learning process and service, learning service availability and transaction support (Dodero et al., 2005).

To represent a learning environment under different complexity levels and mixing didactic techniques, it is necessary for the EML to be expressive enough to specify complex activity structures, dependences, rules, contents, roles, and participants, which can interact to achieve the learning objectives. These structures allow the building of learning flows.

At first, EMLs were thought only for E-Learning Systems containing all the resources the learning scenario will need to guarantee their availability. But as the employed resources evolved, such systems are not able to substitute them in an easy and transparent way, and furthermore, to provide a rich and diverse pedagogical experience for the learner, E-Learning Systems must implement a vast amount of applications –e.g. Moodle¹ forum, wiki, assignment, chat, choice, glossary, lesson, quiz, resource, survey, workshop and other

¹ <http://moodle.org>

modules; or LAMS² chat, forum, multiple choice, noticeboard, notebook, question and answer, survey, voting and other activities—, making even bigger the just mentioned issue. From a software architectural point-of-view, a service-oriented architecture (SOA) (Erl, 2004) provides an important approach in order to implement dynamically learning environments from diverse, distributed and heterogeneous learning resources and services, these can be hardly synchronized and coordinated through an EML specification. Furthermore, based upon SOA principles, Grid-Computing (Foster and Kesselman, 2003) offer a powerful environment to use of resources such as processing power, disk storage, applications and data, allowing the dynamic and unanticipated composition of distributed learning web services (Torres et al., 2008). These orchestration of web services and the long-lived nature of some learning activities, makes necessary for the execution engine to provide transactional support in order to guarantee the reliable execution of activities.

This chapter is organized as follows. Section two presents the four stage evolution of E-Learning Systems, outlining the interest in each stage, which moved from the focus on the development and deployment of Web contents of stage one, to the focus on Complex Learning Processes (CLP) in stage four. Section three presents an explanation of CLP, an educational perspective that combines different ways of learning and is focused on the participative role of the learner in the learning process. Section four presents the characteristics of EMLs and provides an overview of the three broadest languages: IMS LD, LAMS and LPCEL.

Section five presents the characteristics of the Learning Process Engine (LPE) and the following sections will discuss the ways the LPE must provide support for the EML characteristics, specifically for learning flow description, dynamic and unanticipated composition and transaction support. Section six shows the current EML design issues and then evaluates the learning flow expressiveness of current EMLs based on Workflow Control Patterns (WCP). Section seven provides a framework for the dynamic and unanticipated composition of learning services based on SOA and Grid-Computing principles. Section eight presents the importance of transactions along with Advanced Transaction Models (ATM), and then proposes a framework for the LPE to provide transactional support. Finally, in section nine we provide conclusions and further work.

2. Evolution of E-Learning Systems

E-Learning systems have been transcendental in the advance of traditional educational models. Learning technologies have evolved through different stages (Torres et al., 2005), each one with a meaningful progress for the design and display of pedagogical content, increasing functionality, semantic information and flexibility to support the different and complex pedagogical environments.

First Stage— This stage is portrayed by the development and deployment of Web contents. Generally, they are simple applications (e.g. text les, HTML pages, and multimedia resources). The interest here is in the standardization of learning objects to be used in the learning process. Objects are simple instructional contents that can be deployed for learning (e.g. lessons, articles, task assignments, etc.). These objects lack of mechanisms to build learning processes, which are reduced to a sequential itinerary through contents.

² <http://www.lamsfoundation.org>

Second Stage– In this stage the interest moves to the management, location and retrieval of contents. The way to deploy contents is improved and personalized according to the learner, as provided by IMS Learning Information Package Specification (LIP) (IMS, 2001). Technologies for the evaluation and tracking of the learner are developed, as IMS Question & Test Interoperability Specification (QTI) (IMS, 2004). Incipiently there is a chance for the exchange of learning resources among applications through interoperability facilities as IMS Content Packaging (IMS, 2003a). In the pedagogical domain, there is the possibility to design learning processes that are found on basic predefined structures with a sequential, hierarchical style. This is usually achieved by using IMS Simple Sequencing Specification (IMS, 2003c).

Third Stage– The search for greater flexibility and reusability is posed here. Interoperability architectures for learning management applications are introduced. A significant example is the Sharable Content Object Reference Model (SCORM) (ADL, 2004). Some initial efforts can be noticed to bring these technologies closer to a greater didactic flexibility. For instance, the proposal of Integrated E-Learning, developed by the Open University of the Netherlands, provides a vision of E-Learning Systems that support learning processes with pedagogical diversity. These interests are also approached by person-centred e-learning patterns. A step forward is taken from a pedagogical perspective, since it is possible to personalize contents and interactions of learners with their learning process, based upon scenarios, activities, roles and profiles, by using IMS-LD (IMS, 2003b). These structures can be predefined by the instructional designer prior to execution.

Fourth Stage– This stage focuses in Complex Learning Processes (CLP), which facilitate learning in rich and personalized pedagogical environments. We will explain this stage in the following section.

3. Complex Learning Process

A Complex Learning Process (CLP) (Doderio et al., 2005, Torres et al., 2006) is the result of the dynamic and unanticipated integration of mixed pedagogies and resources, and based upon the collaboration of instructors and learners, within a learning process (see Fig. 1). From a constructivist viewpoint, CLP recognize that learning is an active process, which is built upon experiences that are achieved by entwining contents, context, and educational objectives into a learning process. Such complexity depends on three main factors:

The complexity of the learning task per se, which is influenced by the multidisciplinary character of the subject of learning, the duration of the learning process, and the diversity of pedagogical approaches that may be applied.

The degree of autonomy of the learner, when it comes to selecting the learning targets, topics, and expected results.

The design and control of the process after the learning goals have been established. This responsibility can be shared among learners and the instructor, and consists in planning and selecting the activities and outcome products.

Fig. 1. Complex Learning Process

The learning flow of a CLP consists in the composition of complex structures, which are formed by different building blocks (e.g. activities, resources, roles, dependences, restrictions, etc.). These elements interact in a complex and collaborative manner to achieve the learning objectives. Moreover, the composition of such structures cannot be fixed or programmed in design time, but on a dynamic and unanticipated basis. In some cases, the initial form of a CLP must be adapted and redefined as the collaboration of learners and instructors changes. This is usually done in the run time, without missing the achieved state of the learning process.

These issues are not the only contributors to the complexity of a learning process. CLP assumes that the learning resources do not only appear as text, audio, lectures, animations, or documents, but they additionally invoke resources and services to be integrated in learning activities that the learner executes in different learning spaces (formal or informal). CLP allows learners' control over some learning activities (i.e. they can create activities). CLP incorporates different assessments techniques, like traditional tests, portfolio-based, outcome-based, self-assessment, and others. These principles are often identified as constructivist pedagogy and life-long learning.

The design of a CLP usually results in quite complex networks of learning (e.g. when a group of learners works together for projects development or problems solution). Activities of the network are executed by diverse roles who interlace their efforts to achieve the learning objectives.

4. Educational Modelling Languages

Educational Modelling Languages (EML) (Koper, 2001, Rawlings et al., 2002) represent an important approximation to integrate diverse educative aspects, allowing the design and implementation of spaces and activities supported by Information and Communication Technologies. With EML it is possible to integrate, under a learning process, personalized educative material for each student –v.gr. activities, resources, objectives, evaluations, profiles–, promoting the active participation of the student with his learning process. Also, these languages are based in the creation of learning scenarios inside a Unit of Learning (UoL). The different elements implicated in the learning experience –v.gr. roles, objectives, results, activities, resources and services–, are modelled in these units along with the organization mechanisms and coordination of the learning experience –v.gr. activity structures, assignment of the activity structures to roles and rules which determine the material display–. The temporal order in which the various learning activities unfold in a UoL is called learning flow (Koper & Tattersall, 2005).

4.1 Characteristics of Design, Execution and Control of EML

Koper (2001) defines a number of requirements for EMLs, which are: formalization, pedagogical flexibility, explicitly typed learning objects, completeness, reproducibility, personalization, medium neutrality, interoperability and sustainability, compatibility, reusability and life cycle. And due to the complexity levels and the vast didactic techniques, EMLs need to support Complex Learning Process (CLP) learning flows.

Fourth Stage Characterization of EMLs

In order to provide CLP support, new characteristics have been defined by (Dodero et al., 2005): (1) pedagogical diversity: EMLs must be able to guide the composition, execution and control of the learning process including various pedagogical mixings and levels of complexity; (2) learning flow description: EMLs need to be expressive enough to specify complex and dynamic structures (e.g. activities, dependences, rules, contents, roles, scenarios and participants); (3) dynamic and unanticipated composition: in some cases the initial specification of a learning process must be redefined and changed as of the collaboration and negotiation of learners and instructors, usually done in the run time, but the state of the learning process achieved at that point must not be missed; (4) separation of learning process and service: EMLs must contain detailed information to enable the dynamic and run-time access to the required services, i.e. semantic interoperability; (5) learning service availability and containment: suitable descriptions of resources and services are needed to maintain required availability that do not rely on self-containment, with the aim of not losing a chance for the further evolution, substitution and integration of other resources and services; and (6) transaction support: EMLs must be provided with operational transaction support to execute a learning process with the possibility to implement long-run activities.

A comparison of EMLs is presented in Table 1. As it can be seen, most EMLs do not have support for pedagogical aspects such as content cognitive structure, objectives, prerequisites and assessment, as well as the learning process and services. In this context, LAMS, the IMS Global Learning Consortium (<http://www.imsglobal.org>) Learning Design (IMS LD) specification and LPCEL are the broadest languages. The rest of the chapter will focus on these languages.

Implemented Elements	CDF	TeachML	LMML	PML	PALO	Xedu	LPCEL	IMS-LD	LAMS	E2ML
Metadata	+	±	+	±	+	+	+	+	±	±
Content	+	++	++	+	++	+	+	+	+	±
Content Structure	±	±	+	+	+	–	+	–	–	–
Content Cognitive Structure	±	+	+	+	+	+	+	–	–	–
Objectives	–	–	–	–	±	±	+	+	–	++
Prerequisites	–	–	–	–	±	±	+	+	–	+
Expected Results	–	–	–	–	–	±	+	+	–	+
Assessment	–	–	–	–	–	–	+	±	++	±
Learning Process										
Activity Sequence	–	–	–	–	+	+	+	+	+	+
Activity Parallelization	–	–	–	–	–	–	+	+	+	+
Optional Activity Parallelization	–	–	–	–	–	–	+	–	+	+
Conditions	–	–	–	–	–	–	+	+	+	+
Messages	–	–	–	–	–	–	+	+	+	–
Roles	–	–	–	–	–	–	+	+	+	+
Groups	–	–	–	–	–	–	+	–	+	–
Services										
Monitoring	–	–	–	–	–	–	+	+	+	–
Search	–	–	–	–	–	–	+	+	+	–
Communication	–	–	–	–	–	–	+	±	+	–
External Tools	–	–	–	–	–	–	+	±	–	–
Other Proprietary Tools	–	–	–	–	–	–	+	–	+	±

Table 1. Comparison of EMLs.

The IMS LD specification includes a language to describe learning experiences as UoLs, and provides a conceptual framework for the specification of learning process designs, which have to be afterwards deployed onto a computer-based execution engine. The LAMS approach has taken a step further by integrating design and execution of learning activities into the same LD language, design environment and execution engine. LPCEL (Learning Process Execution and Composition Language), provides a framework including the appropriate languages primitives to describe execution-aware learning designs, but still lacks of an implementation.

4.2 IMS Learning Design

IMS LD (Koper & Tattersall, 2005) borrows the theater metaphor to model a teaching-learning process. As illustrated in Fig. 2, the method part is where the top-level coordination of people and activities takes place. A play, as in a theatrical play, consists of acts, although there can be one-act plays. Acts run in sequence, with one starting when the previous act has finished, and the play ends with the completion of the last act. The transition from one act to another serves as a synchronization point for the multiple participants in a learning design, ensuring that they can all start the next act at the same time.

Fig. 2. IMS LD Information Model (Koper & Tattersall, 2005)

Fig. 3. IMS LD Transactions (Torres et al., 2006)

An act is composed of activity, activity structure, and role-part. In every act, the role-part associates a role with an activity (i.e. learning activity, support activity or activity structure). An activity is the basic learning unit and references to a physical learning material. It usually links to contents including plain text, hypertext, graphics, streaming media, and so on. This binding is accomplished through the environment tag for the sake of reusability.

The learning flow (Chen et al., 2005, Torres et al., 2006) of a UoL is generally represented by the nested structure of learning activities depicted in Fig. 3. An UoL is then organized as an activity tree, in which the top level method element puts together multiple concurrent and independent scenarios or plays. Inside a play, an act controls the synchronization of a scenario. A scenario, in turn, can include a number of acts that are executed sequentially. Acts may contain diverse role-parts, which are concurrent and independent, but synchronous. The learning activities of any activity structure under a role-part must be executed sequentially.

IMS LD implements collaboration between roles as the synchronized depiction of a number of activity lanes, which are portrayed independently as individual work for each role that participates in the collaboration. In this model, it is only possible to assign complete activity structures to specific roles, which cannot be changed or be dynamically associated to other activities. Fig. 4 represents the execution of a UoL instance along time.

Fig. 4. IMS LD Execution (Torres et al., 2006)

As a part of a research project at Massey University in New Zealand, Kim Hagen, Diana Hibbert and Kinshuk created Voyager, a prototype web-based learning management system, in compliance with the IMS LD specifications. Perceived benefits of such a system include the flexibility of the IMS LD model for different pedagogical approaches and differing levels of use. However, the educators who created the prototype LD found the theatre metaphor difficult to understand at first, and found it difficult to design a LD with multiple workflows and optional interactions. It was also difficult to model activities which are not done in a particular sequence, for example, learners completing an assessment activity may do it before or after attending class (another activity). For learners, the theatre metaphor introduces a new model which they must learn in order to learn the course content, increasing their cognitive load (Hagen et al., 2006).

4.3 LAMS

The Learning Activity Management System (LAMS) (Dalziel, 2003) is a software system capable to design, manage and deploy learning scenarios employing Internet based technologies. For that, LAMS integrates environments for user administration, student run-time delivery of sequences, teacher run-time monitoring of student sequences and, most importantly, teacher authoring/adaptation of sequences.

In LAMS, the learning design is centered in the design of a sequence of learning activities, which students need to carry out individually or in small groups to achieve the learning objectives. Each activity includes the tools and resources needed, such as: question and answer (with student answers shared with the group either anonymously or identified), polling (with total responses shared with the group), asynchronous discussion forum, synchronous chat, noticeboard (simple text content/instructions), resource presentation and sharing (URL/webpages/files), notebook/journal, assessment submission, MCQ and True/False (with options to display feedback, average class score and “high” score), and various combinations of tools, including “chat and scribe”. In addition, a grouping tool allows any tool to run in either whole class or small group modes. The flexibility of the learning design indicates that many more collaborative tools can be developed to broaden the impact of the LAMS platform.

Fig. 5. LAMS Learning Design Data Model

LAMS was inspired by IMS Learning Design. However, LAMS was not designed to be a reference implementation of the specification, but it is expected to be IMS Learning Design conformant as both LAMS and IMS LD evolve to address current IMS LD implementation challenges. Despite that LAMS does not propose an EML and its limited use of standards

and specifications, the LAMS Learning Design Data Model presented in Fig. 5, suggests a practical orientation over some aspects barely boarded in current EMLs. Just to mention an example, LAMS maintains a special interest in learning design based in individual and collaborative activities; also an effort is made in the development of tools to facilitate the performing of such activities. And finally, LAMS incorporates some flexibility to design activity structures besides simple sequence of activities, such as, parallel activities, optional activities and synchronization mechanisms.

4.4 LPCEL

The Learning Process Composition and Execution Language (LPCEL) (Dodero et al., 2005, Torres et al., 2006) is a formal language that provides a containment framework of elements that can describe any design of a CLP. LPCEL is used to specify complex and dynamic activity structures, including roles, groups, dependences, restrictions, scenarios, contents, services, assessments and other resources, which interact to achieve a set of learning objectives. LPCEL allows re-designing the learning process in a dynamic and unanticipated fashion. LPCEL can be used to describe the execution of a learning process and to orchestrate the services and resources that are needed to carry it out. LPCEL can also integrate external applications and services into a learning process, so it becomes the core element of a service-oriented architecture that is thought to integrate, compose, and execute a CLP.

Fig. 6. LPCEL Information Model

A conceptual overview of LPCEL is shown in Fig. 6. This model has four main groups of elements:

General information of the process, including prerequisites, learning objectives and/or outcomes, and general features.

The structure of the process, formed by complex activities, basic activities and roles.

The detailed activities that take part in the process, i.e. learning activities, assessment activities and context activities.

The resources and services that can be sought accessed, invoked, or composed by each activity.

The `<lpcel-definition>` element allows defining diverse `<complexlearningprocess>` to represents a CLP. Each one of these composed of one or several `<component-clp>` to design complex learning structures. A `<complexcomponent>` includes one or several `<basic-component>` and/or `<complexcomponent>`. Also it makes reference to a collection of elements to define learning flow (e.g. `<complex-component>`, `<parallel>`, `<join>`, `<until>`, `<foreach>`), including their relationships and a number of criteria to indicate the start and termination conditions of the `<complex-component>`. A `<component-activity>` contains a collection of elements to execute activities like `<learningactivity>`, `<assessmentactivity>` or `<contextactivity>`. Each basic activity (i.e. learning, assessment or context) can refer to different specifications of `<resource>` to be used in the activity. The `<resource>` can invoke local (i.e. `<local-access>`) or remote resources (`<remote-access>`) such as SCORM contents or remote applications (e.g. a project repository, virtual laboratories, simulators and collaborative tools, and design tools among others services). The `<role>` and `<permission>` are assigned to `<componentactivity>` and `<complex-component>` (i.e. role assignment). Moreover, the `<aim-prerequisite>` contain a sequence of elements for the definition of the `<prerequisite>`, `<outcome>` and `<learning-objective>` that can be assigned to complex and basic activities.

5. Fourth Generation Learning Process Engines

As seen in previous the section, EMLs are used to describe learning scenarios. But as in workflow management systems where the execution is in charge of the workflow engine (Hollingsworth, 1995), the Learning Process Engine (LPE) is the software responsible for (1) interpretation of the process definition; (2) control of process instances as creation, activation, suspension, termination; (3) navigation between process activities as sequential or parallel operations, deadline scheduling, interpretation of workflow relevant data and others; (4) sign-on and sign-off of specific participants; (5) identification of work items for user attention and an interface to support user interactions; (6) maintenance of learning flow control data and learning flow relevant data, passing learning flow relevant data to/from applications or users; (7) an interface to invoke external applications and services, and link any learning flow relevant data; and (8) supervisory actions for control, administration and audit purposes.

In workflows, many of these features are often handled by the means of implementing an Advanced Transaction Model(ATM) to ensure correctness and reliability. When this is the case they are called Transactional Workflows. In such a workflow, the process activities are mapped to constituent transactions of an advanced transaction supported by an ATM, and

control flow is defined as dependencies between transactional steps. An extra control layer in terms of dependencies is added to ATM to provide functionality to the transactions running in large-scale distributed information systems environment. But it is necessary to remark transactional features form only a small part of a large-scale workflow application. Workflow requirements either exceed, or significantly differ from those of ATMs in terms of modelling, coordination and run-time requirements. So it is definitely useful to incorporate transactional semantics such as recovery, relaxed atomicity and isolation to ensure reliable workflow executions. Nevertheless, to view a workflow as an ATM, or to use existing ATMs to completely model workflows would be inappropriate (Worah and Sheth, 1997).

Fourth generation LPEs, must be able to support all the new EML characteristics mentioned in the previous section. The rest of the chapter will focus on the way LPEs should provide the support for these characteristics, focusing on: learning flow description, dynamic and unanticipated composition and transaction support.

6. Learning Flow Description

EMLs such as IMS LD, LAMS and LPCEL can be used to describe a learning process, but they depict different degrees of expressiveness to represent learning flows. When working with UoLs, the design phase is fundamental because an UoL must be fully designed before it is delivered. The aspiration of this technology is to develop knowledge, abilities and competences employing instruction under several paradigms (Jochemes et al., 2004, Koper, 2003, Rosmalen et al., 2003).

Defining all activities and assign users at design time will result in a too rigid scheme that cannot be adapted to the concrete events coming out from executing the learning sequences. On the contrary, leaving all activities and task assignments to the run time, the result will be a more flexible scheme, but with no learning design at all. LAMS, IMS LD and LPCEL adopt a mixed approach. They allow defining some elements at design time, and letting other elements at run time. For instance, LAMS gates and grouping activities, or IMS LD Level B elements, are used to set up design-time solutions prepared to manage run-time issues, such as binding user groups to running activities. The LPCEL language, can manifest these execution-related issues emerged in a process-oriented Learning Design (Dodero and Torres, 2008).

6.1 Learning Process Design Issues

Learning process design issues to be considered during execution are:

- (1) Composite Activities: When designing a learning process, pattern-based groups of activities that can be sequenced, selected and concurrently executed, are often needed. In IMS LD, activity-structures can be used, but they can be only sequenced or selected. If concurrent activities are needed, they can be defined at the play level. An alternative is defining different role-parts within the same act, which does not make sense when a single learner is executing the activities. A third alternative is managing activities' completion status properties, but this is too burdensome when a large number of activities are considered. LAMS provides sequenced, parallel, and optional activities that comprise the complex activity type, which makes easier to describe such concurrent composite activities.
- (2) Conditional Branching and Looping: The usual learning flow of activities must be branched sometimes, according to a special condition that often depends on the learner

profile or the results of previously executed activities. Other times, remedial learning activities may require executing an activity or activity group an indefinite number of times, until the remediation condition is achieved by the learners. With IMS LD, to branch an activity an ad hoc property and a set of conditions must be defined to drive the learning flow through the branched activities. When LAMS is used, conditional branching is expressed through branching activities, which drive the learning flow depending on the teacher selection, grouping objects, or the output of previous tool-based activities.

(3) Parallel Branching on Users' Learning Flows: Sometimes is needed to fork the learning flow of one learner, a group of learners, or all learners, to define a branching of parallel but different activities, which are executed by different groups of learners. This is achieved in IMS LD by defining roles or properties plus conditions, but once the execution of an IMS LD run has started, the association of users to roles cannot change. On the contrary, LAMS provides a grouping activity type that permits expressing that association during run time. The combination of a LAMS branching activity with a grouping activity enables forking user groups on a set of activities during run time, without requiring defining additional properties and conditions.

(4) Synchronization points: Setting some synchronization points can be useful in a learning flow of activities, when parallel running activities must join on a given instant before proceeding with the following ones. The IMS LD specification states that plays always run concurrently, whilst acts always run in sequence. Synchronization points can be only marked on completion of the activities, according to user choices or time limits, or when a property is set. With LAMS, gate activities can be used to synchronize learning flows of activities, holding concrete semantics of concurrent access.

(5) Split running activities: A special case of learning flow synchronization occurs when two flows of activities (possibly run by different users) must synchronize in certain points along a number of iterations. IMS LD does not provide adequate primitives to describe this pattern. In LAMS gates and flows can be used to describe that complex structure, although it is not a straightforward learning design task.

6.2 Workflow Control Patterns

Workflow Control Patterns (WCP) (Russell et al., 2006) have been identified with the aim of delineating the fundamental requirements that arise during business process modelling on a recurring basis and to describe them in an imperative way. So far, 43 WCP have been identified and classified in Basic Control Flow, Advanced Branching and Synchronization, Multiple Instance, State Based, Cancellation and Force Completion, Iteration, Termination, Trigger and Disclaimer. A comprehensive description of a workflow process also requires consideration of the data perspective. With this in mind, another 40 patterns in the data perspective have been identified.

Learning flows range from basic activity structures such as sequence, parallel, loops, split and union; up to more complex structures such as multiple choices and multiple unions. In this section, an EML expressiveness evaluation will be presented focusing on 10 WCP including all the Basic Control Flow Patterns and some of the Advanced Branching and Synchronization and the Iteration Patterns, because of their capability to represent the most common learning scenarios.

Basic Control Flow patterns capture elementary aspects of process control, namely: (1) Sequence: an activity in a workflow process is enabled after the completion of a preceding

activity in the same process; (2) Parallel Split: the divergence of a branch into two or more parallel branches each of which execute concurrently; (3) Synchronization: the convergence of two or more branches into a single subsequent branch such that the thread of control is passed to the subsequent branch when all input branches have been enabled; (4) Exclusive Choice: the divergence of a branch into two or more branches, such that when the incoming branch is enabled, the thread of control is immediately passed to precisely one of the outgoing branches based on the outcome of a logical expression associated with the branch; and (5) Simple Merge: the convergence of two or more branches into a single subsequent branch. Each time an incoming branch is enabled, results in the thread of control being passed to the subsequent branch.

Advanced Branching and Synchronization patterns characterize more complex branching and merging concepts. Although relatively commonplace in practice, these patterns are often not directly supported or even able to be represented in many commercial offerings. The patterns of this category included in the evaluation are: (1) Multi-Choice: the divergence of a branch into two or more branches, such that when the incoming branch is enabled, the thread of control is passed to one or more of the outgoing branches based on the outcome of distinct logical expressions associated with each of the branches; (2) Structured Synchronizing Merge: the convergence of two or more branches (which diverged earlier in the process at a uniquely identifiable point) into a single subsequent branch, such that the thread of control is passed to the subsequent branch when each active incoming branch has been enabled; (3) Multi-Merge: the convergence of two or more branches into a single subsequent branch. Each enablement of an incoming branch results in the thread of control being passed to the subsequent branch; and (4) Structured Discriminator: the convergence of two or more branches into a single subsequent branch following a corresponding divergence earlier in the process model. The thread of control is passed to the subsequent branch when the first incoming branch has been enabled. Subsequent times incoming branches that are enabled do not result in the thread of control being passed on. Finally the discriminator construct is reset when all incoming branches have been enabled.

Iteration patterns deal with capturing repetitive behavior in a workflow. Structured Loop is the Iteration pattern included in the evaluation, and it is described as the ability to execute an activity or sub-process repeatedly. The loop has either a pre-test or post-test condition associated with it that is either evaluated at the beginning or end of the loop to determine whether it should continue. The looping structure has a single entry and a single exit point.

6.3 Workflow Control Patterns and EML

Although WCP provide a good point of reference to design learning flows and evaluate the expressiveness of EMLs, it is important to differentiate between learning and workflows. In a learning environment the execution of a process activity does not guarantee that the objectives have been achieved, meanwhile in workflows, the execution of a process activity assures a state change. In other words, the completion of a learning activity does not assure new knowledge for the learner.

An EMLs learning flow expressiveness evaluation based on the WCP previously described is presented in Table 2. IMS LD only has full support for the sequence pattern. A parallel split can be achieved by defining more than one play due to the concurrence and independence of them, but this will lead to a loss of sense of the script metaphor, because a play means a teaching-learning process and there will be many learning processes were

pedagogically only one is supposed to be. Also there will be no synchronization for the plays later on. Also, IMS-LD provides the option to choose a number of activities. This does not mean the Exclusive Choice, Multi-Choice, Synchronization, Simple Merge, Multi-Merge or Structured Discriminator patterns are supported, because activities are then executed in a rigid sequence manner.

Workflow Control Pattern	IMS-LD	LAMS	LPCEL
Basic Control Flow			
1. Sequence	+	+	+
2. Parallel Split	±	±	+
3. Synchronization	–	±	+
4. Exclusive Choice	–	–	+
5. Simple Merge	–	±	+
Advanced Branching and Synchronization			
6. Multi-Choice	–	–	+
7. Structured Synchronizing Merge	–	–	+
8. Multi-Merge	–	–	+
9. Structured Discriminator	–	–	+
Iteration			
10. Structured Loop	–	–	+

Table 2. EMLs learning flow expressiveness evaluation

In the LAMS Learning Design Data Model, full support for the Sequence pattern through “SequenceActivity” is provided. The Parallel Split, Synchronization and Simple Merge patterns are supported but limited to a set of activities already defined. The support for Synchronization and Simple Merge is provided by gateways such as “SynchGateActivity”, “ScheduleGateActivity” and “PermissionGateActivity”. A Parallel Split is achieved by the “ParallelActivity” element. The situation for the rest of the patterns, is the same as in IMS LD.

LPCEL, to define learning flow makes reference to a collection of <Complex-Component> elements such as <Sequence>, <Parallel>, <Choice>, <Switch>, <While>, <doWhile>, <Join>, <Split> and <CompensateCA>; and also to a collection of <Basic-Component> elements such as <Action>, <Flow>, <Delay>, <Invoke>, <Gateway>, <Terminate>, <Assig>, <Create> and <CompensateBA>; including their relationships and a number of criteria to indicate the start and termination conditions of the <Component-CLP>, providing support for a vast number of WCP including all the basic ones, as seen in table 2 but not limited only to them. These primitives allow LPCEL to support WCP, providing the necessary expressiveness to describe and guide the execution of a CLP.

7. Dynamic and Unanticipated Composition

Inside a CLP environment, there is a great variety of resources and services in which learners can access to achieve the learning objectives, some of them have a direct implication on learning itself –learning services and resources–; others allow to generate spaces for the development of collective competences –community services and resources–; and others show the activity students carry out in the learning environment –context services and resources–. Some examples for each category are: (1) Learning services and resources: Virtual labs, tutorials, simulators, educative contents, assessment systems, search systems, software with particular educative objectives e.g. programming environments, design environments, math and statistics tools, business games, and others. (2) Community services and resources: Collaborative tools, communication services, forums, distribution lists, coordination tools, shared agendas, repositories for group work or document management and others. (3) Context services and resources: Tutoring services, digital library services, university services, and others.

Some of these services may be recovered and deployed locally, and other may be executed in a distributed way, based in a Service-Oriented-Architecture (SOA) (Erl, 2004), also allowing the integration of new resources and services to the learning process.

7.1 Grid-Services within SOA

Grid networks consist in large-scale distributed systems that share physical heterogeneous resources (e.g. computing, storage, network components and equipment, sensors, etc.), and make possible the creation of virtual organizations (Foster and Kesselman, 2003). Furthermore, these capabilities enable powerful, flexible, pervasive, and cost-effective services to the users. Initially, Grid network services were focused on the “Anatomy” (e.g. computing power, data storage and network bandwidth), currently, it is recognized that for effective resource sharing and virtual communities support, the “Physiology” is also necessary – A SOA approach.

In such a highly distributed community, the orchestration and management of numerous distributed hardware and software components presents a big challenge. Therefore, a service-oriented infrastructure was introduced to the resource management and provisioning mechanisms used to meet quality of service goals. Grid services come to serve this purpose; Grid services are essentially Web services with specific extensions or interfaces for use in Grids (Pankatrius and Vossen, 2003). A Grid service meets a set of conventions for service lifetime management, inspection, and notification of changes in service states. Grid services provide for the controlled management of the distributed and often long-lived state commonly required in distributed applications.

Grid networks and services could help the learning to happen in different network domains and to enable high-performance experiments and applications to be used in learning experiences. Through the adoption of these technologies it is possible a wide-scale learning resource sharing in heterogeneous and geographically distributed environments, the implementation of learning organizations in which different actors (universities, teachers, learners) share a common target and are able to cooperate to get the same goal.

The integration of Grid services within the SOA framework provides more flexibility and make possible to leverage on the resources distributed across Grid networks in order to build the learning experience that best fit student requirements.

7.2 Web Applications and Services Enhanced Learning Architecture

LPCEL and the Web Applications and Services Enhanced Learning Architecture (see Fig. 7) are the first approach to the fourth stage of E-Learning Systems. In this proposed architecture, the learning scenario is designed with the Learning Process Editor and executed in the Learning Process Engine, which through semantic web services such as WSDL and WADL communicates with the Learning Services Bus where the Learning Web Services are plugged in. User interface must be Web 2.0-enabled and communication is made through an API WS-Access. Interoperability with other Learning Management Systems such as Moodle or LAMS can be achieved by Wrap ad-hocs.

Fig. 7. Web Applications and Services Enhanced Learning Architecture

8. Transactional Support

A transaction (Gray and Reuter, 1993) is a consistent and reliable computational unit, which executes from an initial consistent state and finishes its execution in a final consistent state, i.e. a transaction is a collection of actions which make a consistent transformation of the states of a system. To achieve this, transactions exhibit the ACID properties: (1) Atomicity: A transaction is executed completely (commits) or the final effect is as it has never been executed (aborts); (2) Consistency: The code of the transaction should guarantee that if applied to a consistent state of the data it should bring the data to a new consistent state; (3) Isolation: The execution of concurrent transactions accessing common data should be equivalent to a serial execution of them; (4) Durability: The effects of a committed transaction should be preserved even in the advent of failures.

To guarantee the execution of a learning flow in EML engines, it is necessary for them to provide transaction support through Advanced Transaction Models(ATMs) to relax the basic ACID transaction properties. Nowadays, there are several ATMs summarized in (Elmagarmid, 1992) that could be suitable for the new generation of EML engines like Nested transactions, SAGAS, Open nested transactions, Flexible transactions, ConTracts, S-Transactions, the DOM Transaction Model and Multi-colored transactions. A comparison between these ATMs is made in Table 4, showing the general purpose of each ATM, the ACID property they relax and a short description of them.

ATM	General purpose	Based on	Relaxed property	Short description
Nested transactions	Preserve all ACID properties for top level transactions		Durability on subtransactions	New transactions (subtransactions) can be started within another transaction yielding to a transaction tree structure.
SAGAS	Long-lived transactions		Isolation	Each subtransaction has associated a compensating transaction. A Saga commits if all its subtransactions also commit.
Open nested transactions	Long-lived activities	Nested transactions	Isolation	Open subtransactions externalise their results. If a transaction aborts, its committed open nested subtransactions are not automatically undone neither compensated.
Flexible transactions	Long-lived activities	Nested transactions	Atomicity and isolation	Two-level nested transactions with three kinds of transactions: compensatable, retrievable and pivot.
ConTracts	Long-lived computations in non-standard applications like workflow and CAD		Isolation	Fault-tolerant execution of a sequence of actions (steps) according to a control flow description (script).
S-Transactions	Cooperation between organizations	Nested transactions	Isolation	Dynamically generated hierarchical distributed transactions. Sub-s-transactions commit as soon as they finish. A transaction decides on its result based on the results of its subtransactions.
The DOM Transaction Model	Long running activities and the execution of system-triggered activities	Nested transactions	Isolation	Nested transactions defined by the user or being the result from the firing of rules while executing a transaction. The subtransaction can be immediate, deferred or detached.
Multi-coloured transactions	Long-lived computations	Nested transactions	Atomicity	Increase the concurrency of nested transactions and allow partial results to became permanent. Three types of subtransactions: Subtransactions of a serializing action, glued actions and top-level independent actions.

Table 4. Advanced Transactional Models

8.1 Advanced Transactional Models in a Complex Learning Process

So, why is this comparison made about the different ATMs? Because each learning activity is different and may require a certain ACID property not to be relaxed, or may have some specific requirements not suitable for every ATM. So it is not enough for the engine to have an ATM implemented, consequently the engine must support different ATMs so different learning activities may perform their transactions based on different ATM models. In general, the proposal is to manage three transactional levels (see Fig. 8) in a LPE:

Process: The general ATM used at the top level of the CLP, specifically on LPCEL, the <Complex-Learning-Process> element. An ATM focused on long-lived activities is suitable for this level.

Activity or subprocess: The most critical transactional level in terms of learning. By choosing the right ATM for each learning activity, a contribution to the achievement of the learning objectives –which are the base of the CLP on LPCEL– is made by the means of providing the right learning flow structure and behavior to the learner. Different ATMs can be used on different activities and structures, e.g. ConTracts may be used on the <Complex-Component> elements involving <Sequence>, <doWhile> and <While> elements or activities following a behaviorism approach, meanwhile S-Transactions can be used for <Complex-Component> elements with the <Switch> and <Split> elements or activities following a constructivism approach.

Resource: This level occurs at the <Resources> element on LPCEL. The main matter is about state and stateless services. With stateless services a simple transactional model with

preserving all the ACID properties can be used, meanwhile with state services the transactional model to use is up to the kind of resource.

Fig. 8. Transactional Levels for a Learning Process Engine

In other words, suppose Alice and Bob are enrolled in a course. The whole course can be viewed as transaction with many subtransactions (learning activities) and because the course would last a considerable amount of time, it is a long-lived activity with partial results where Open nested transactions can be used as the Process level ATM. But then, what happens when the professor wants his students to develop certain skills which can be better obtained through a behaviorism approach, where a sequence of tasks must be completed in a certain order until each task is fully completed in an everything or nothing basis? A ConTract ATM for the Activity level must be implemented for it. And then, when it is time for a collaborative activity with a constructivism approach, where Alice and Bob need to be working together with partial deliverables to achieve a certain objective, the S-Transactions model is more suitable. And finally, each task has associated resources to it, some of them could be simple stateless services which can be handled by the Activity level ATM or perhaps by a simple transactional model with all the ACID properties, but others may need complex interaction with state services, if this is the case, a different ATM may be implemented depending on the kind of resource.

On learning flow activities, when a transaction fails can be because a software or hardware such as unavailability of resources, incorrect input formats, internal application failures and others; or also it could be because the learner did not achieved the learning objectives, even though the activity was completed, i.e. a grade below the minimum to pass a test, a correction of a stakeholder in a project, and others. So a LPE has to provide the ways to deal with learning flow compensation activities to achieve the learning objectives of each task.

9. Conclusion

The enhanced set of characteristics and flexibility offered by the e-learning systems evolution provides new and better ways to facilitate learning, particularly, the fourth stage service oriented approach. In this perspective, it is of high relevance to show our vision of a new generation of learning web services, which need to be integrated and executed by the participants in a pedagogical sense, *i.e.* participants may perform learning activities by consuming many different web services. Some services have already been developed by students, faculty and staff from the Tecnológico de Monterrey. The following are some examples:

WS-TestMobile: A web service implementation of the IMS Question and Test Interoperability Standard (IMS, 2004). WS-Rubrics: A web service for the elaboration and application of rubrics. WS-Qualitative Research: A web service to provide support for qualitative research techniques. WS-Wiki: A web service for the elaboration of wikis. WS-Collaborative Lecture: A web service to download lectures and make comments in a collaborative way. WS-Project Management: A web service interface for the Trac System³. WS-Brainstorming: A web service to provide support for the brainstorming technique.

As web services, they can be consumed by different clients, *e.g.* web clients or even a mobile device client, allowing the participant not to be attached to a computer terminal. Also, services can be located in different places or even replicated for grid computing.

In most of cases, activities are not performed in an anarchic way. For that, it is necessary to employ EMLs able to orchestrate and drive the learning process of each participant. As an example, with these specific services and LPCEL, it is possible to design a learning scenario starting with a collaborative lecture and followed by a brainstorming of project ideas. Once the project is chosen, WS-Project Management, WS-Qualitative Research and WS-Wiki, could be used in parallel and in an iterative and collaborative way to do the project. When the project is finished and presented, it is evaluated through a rubric and a self-evaluation by employing WS-Rubrics and WS-TestMobile.

Service-Oriented Learning Process Engines, represent a new generation of e-learning systems. From a technological and pedagogical perspective, it is relevant the study of ATMs in order to guarantee a reliable execution of a learning scenario. Finally, from a technological and communications perspective, SOA represents an opportunity for the implementation of these systems. Further work consists in testing the current developed web services in real scenarios.

10. References

- ADL (2004) *ADL. SCORM: Content Aggregation Model (CAM) Version 1.3*. Technical report, Advanced Distributed Learning (ADL), 2004.
- Chen, M.; Chen, C.; Cheng, Y. C. & Hsieh, C. (2005). On the development and implementation of a sequencing engine for IMS Learning Design specification. In *Advanced Learning Technologies, 2005. ICAALT 2005. Fifth IEEE International Conference on*, pages 636-640, ISBN: 0-7695-2338-2, July 2005, Kaohsiung, Taiwan.

³ <http://trac.edgewall.org/>

- Dalziel, J. R. (2003). Implementing Learning Design: The Learning Activity Management System (LAMS). In G. Crisp, D. Thiele, I. Scholten, S. Barker, and J. Baron, editors, *Proceedings of the 20th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education (ASCILITE)*, ISBN: 0-9751-7020-1, Adelaide, Australia, December 2003.
- Dodero, J. M. & Torres, J. (2008). Awareness of execution in designing learning activities with LAMS and IMS LD. In *European LAMS Conference 2008*, ISBN: 978-1-74138-287-7, Cadiz, Spain, June 2008.
- Dodero, J. M.; Torres, J.; Aedo, I. & Díaz P. (2005). Beyond descriptive eml: Taking control of the execution of complex learning processes. In *Simposio Pluridisciplinar sobre Diseño, Evaluación y Descripción de Contenidos Educativos Reutilizables (SPDECE)*, ISBN: 84-9788-350-0, October 2005.
- Elmagarmid, A. (1992), editor. *Database Transaction Models for Advanced Applications*. Morgan Kaufmann, ISBN: 1558602143.
- Erl, T. (2004). *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall PTR, ISBN: 0-13-142898-5.
- Foster, I. & Kesselman, C. (2003). *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, ISBN: 1-55-860933-4.
- Gray, J. & Reuter, A. (1993). *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, ISBN: 1-55-860190-2.
- Hagen, K.; Hibbert, D. & Kinshuk, P. (2006). Developing a learning management system based on the IMS Learning Design specification. In *Advanced Learning Technologies, 2006. ICAIT 2006. Sixth IEEE International Conference on*, pages 420-424, ISBN: 0769526322, July 2006, Kerkrade, The Netherlands.
- Hollingsworth, D. (1995). *The workflow reference model – issue 1.1. Specification TC00-1003*, Workflow Management Coalition, 2 Crown Walk, Winchester, Hampshire, UK, SO22 5XE, Jan 1995.
- IMS (2001). *IMS Learner Information Packaging Information Model Specification, Version 1.0 Final Specification*. Technical report, IMS Global Learning Consortium, March 2001.
- IMS (2003a). *IMS Content Packaging Information Model, Version 1.1.3 Final Specification*. Technical report, IMS Global Learning Consortium, June 2003.
- IMS (2003b). *IMS Learning Design Information Model, Version 1.0 Final Specification*. Technical report, IMS Global Learning Consortium, January 2003.
- IMS (2003c). *IMS Simple Sequencing Information and Behavior Model, Version 1.0 Final Specification*. Technical report, IMS Global Learning Consortium, March 2003.
- IMS (2004). *IMS Question and Test Interoperability: Item Overview, Version 2.0 Public Draft*. Technical report, IMS Global Learning Consortium, June 2004.
- Jochemes, W.; Van Merriënboer, J. & Koper, R. (2004). An introduction to integrated e-learning. In *Integrated E-Learning: Implications for pedagogy, technology & organization*. Morgan Kaufmann, ISBN: 0-415-33502-7, San Mateo, CA.
- Koper, R. (2001). *Modeling units of study from a pedagogical perspective: the pedagogical metamodel behind eml*. Technical report, Open University of the Netherlands, June 2001.
- Koper, R. (2003). Chapter 6. In *Learning technologies: An integrated domain model*, pages 64-79. RoutledgeFalmer, London.

- Koper, R. & Tattersall C. (2005). *Learning Design, A Handbook on Modelling and Delivering Networked Education and Training*. Springer, ISBN: 3-540-22814-4, Berlin Heidelberg New York.
- Pankatrius, V. & Vossen, G. (2003). Towards e-learning grids: Using grid computing in electronic learning. In *IEEE Workshop on Knowledge Grid and Grid Intelligence*, pages 4-15, Halifax, New Scotia, Canada, 2003.
- Rawlings, A.; Van Rosmalen, P.; Koper, R.; Rodríguez-Artacho, M. & Lefrere, P. (2002). *Survey of Educational Modelling Languages*. Technical report, CEN/ISSS WS/LT, September 2002.
- Van Rosmalen, P.; Brouns, F.; Tattersall C.; Vogten, H.; Van Bruggen, J. & Koper, R. (2003). Towards an open framework for adaptive, agent-supported e-learning. In *International Journal of Continuing Engineering Education and Life-Long Learning*, vol. 15, no3-6, pp. 261-275, ISSN: 1560-4624.
- Russell, N.; Ter Hofstede, A.H.M.; Van Der Aalst, W.M.P. & Mulyar, N. (2006) *Workflow Control-Flow patterns: A revised view*. Technical report, BPM Center, 2006.
- Torres, J; Dodero, J. M.; Aedo, I. & Días-Pérez, P. (2005). A characterization of composition and execution languages for complex learning processes. In *Proceedings of the 4th IASTED International Conference on Web-Based Education (WBE 2005)*, ISBN: 0-88986-480-2, , Grindelwald, Switzerland, February 2005.
- Torres, J.; Dodero, J.M.; Aedo, I. & Diaz, P. (2006). Designing the execution of learning activities in complex learning processes using LPCEL. In *Advanced Learning Technologies, 2006. ICALT 2006. Sixth IEEE International Conference on*, pages 415-419, ISBN: 0769526322, July 2006, Kerkrade, The Netherlands.
- Torres, J.; Cardenas, C.; Dodero, J. M. & Aedo, I. (2008). A grid-based architectural framework for composition and execution of complex learning processes. In *Advanced Learning Technologies, 2008. ICALT 2008. Eighth IEEE International Conference on*, pages 61-63, ISBN: 978-0-7695-3167-0, July 2008, Santander, Cantabria, Spain.
- Worah, D. & Sheth, A. (1997). Transactions in Transactional Workflows. In *Advanced Transaction Models and Architectures*, Jajodia, S. & Kershberg, L., (Ed.), pages 3-45. Kluwer Academic Publishers, ISBN: 0-7923-9880-7, Norwell, MA, USA.

IntechOpen

Advances in Learning Processes

Edited by Mary Beth Rosson

ISBN 978-953-7619-56-5

Hard cover, 284 pages

Publisher InTech

Published online 01, January, 2010

Published in print edition January, 2010

Readers will find several papers that address high-level issues in the use of technology in education, for example architecture and design frameworks for building online education materials or tools. Several other chapters report novel approaches to intelligent tutors or adaptive systems in educational settings. A number of chapters consider many roles for social computing in education, from simple computer-mediated communication support to more extensive community-building frameworks and tools. Finally, several chapters report state-of-the-art results in tools that can be used to assist educators in critical tasks such as content presentation and grading.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Jorge Torres, Cesar Cardenas, Juan Manuel Dodero and Eduardo Juarez (2010). Educational Modelling Languages and Service-Oriented Learning Process Engines, *Advances in Learning Processes*, Mary Beth Rosson (Ed.), ISBN: 978-953-7619-56-5, InTech, Available from: <http://www.intechopen.com/books/advances-in-learning-processes/educational-modelling-languages-and-service-oriented-learning-process-engines>

INTECH
open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](https://creativecommons.org/licenses/by-nc-sa/3.0/), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen