ADVANCED TRANSACTIONAL MODELS FOR A NEW GENERATION OF EDUCATIONAL MODELLING LANGUAGE ENGINES

Jorge Torres, Eduardo Juárez
Departamento de Sistemas Computacionales
Tecnológico de Monterrey, México
{jtorresj, A00339777}@itesm.mx

Juan Manuel Dodero
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Cádiz, Spain
juanma.dodero@uca.es

Ignacio Aedo
Departamento de Informática
Universidad Carlos III de Madrid, Spain
aedo@ia.uc3m.es

Abstract

E-Learning Systems have evolved moving from self-contained simple static contents, to the design and execution of learning scenarios described by Educational Modelling Languages (EML) and the integration of web services. But also carrying with implications such as dealing with long-lived learning activities and finishing activities without accomplishing their learning objectives. Advanced Transactional Models (ATM) relax some of the basic ACID transaction properties so applications may handle in a reliable way, several types of long-lived activities and compensation activities. An EML execution engine must implement different ATMs in order to provide different levels of transaction support to deal with the diverse types of learning activities.

1 Introduction

E-Learning systems have been transcendental in the advance of traditional educational models. Learning technologies have evolved through different stages [1], starting from simple static Web contents, and moving to the standardization of learning objects, giving place to the creation of Educational Modelling Languages (EML) able to fully describe learning scenarios. At first, EMLs were thought only for E-Learning Systems containing all the resources the learning scenario will need guaranteeing 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, making even bigger the just mentioned issue. To aboard this issue, the next stage of EML engines must provide facilities to handle: (1) interpretation of the learning 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 workitems 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. So it is definitively 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.

3 Advanced Transactional Models for Complex Learning Processes

A transaction [6] is a consistent and reliable computational unit, which executes from an initial consistent state and finishes its execution in a final consistent state. To achieve this, transactions exhibit the ACID properties (Atomicity, Consistency, Isolation and Durability). To guarantee the execution of a learning flow in EML engines, it is necessary for them to provide transaction support through ATMs to relax the basic ACID transaction properties.

Considering the learning environment characteristics, it is also necessary to mark the difference between a learning flow activity and a workflow activity. Workflow activities are executed and when the transaction fails it is due solely to a software or hardware problem, such as unavailability of resources, incorrect input formats or internal application failures. On learning flow activities, when a transaction fails can be due also because the learner did not achieve the activity objectives, even though the activity was completed. So EML engines need to provide the ways to deal with learning flow compensation activities to achieve the learning objectives of each task.

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 an 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 in EML engines: (1) Process: The general ATM used at the top level of the CLP, specifically in LPCEL, the <Complex-Learning-Process> element. An ATM focused on long-lived activities is suitable for this level. (2) 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 in 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. (3) 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 preserving all the ACID properties can be used, meanwhile with state services the transactional model to use is up to the kind of resource.

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.

4 Conclusions

Learning activities are vast, diverse and may be long-lived. They employ different methods and psychological approaches, so it would be rigid to execute all of them under a single transactional model, which would lead to not achieve the learning objectives they are intended to, resulting in technology being an obstacle rather than a booster.

5 Acknowledgements

The work is partly funded by the Avanza subprogramme (project TSI-020501-2008-53), the PCI subprogramme (project A/018126/08) of the Spanish Government and the Distributed and Adaptive Systems Lab for Learning Technologies Development, DASL4LT (C-QRO-17/07) from the Tecnológico de Monterrey, México.