Chapter 2.14
SOA-Frameworks for Modular Virtual Learning Environments: Comparing Implementation Approaches

Fredrik Paulsson
Umeå University, IML

Mikael Berglund
Umeå University, Ladok

ABSTRACT

A general SOA framework for Virtual Learning Environments, based on the VWE Learning Object Taxonomy, is suggested in this chapter. Five basic and general services are suggested for implementation of modular Virtual Learning Environments. The design of the service framework was tested by implementation in two prototypes, using two different approaches where a Java-RMI based implementation was compared to a Web Service (SOAP) based implementation. By implementing the VWE Learning Object Taxonomy and the VWE SOA framework, the prototypes showed that a level of modularity, similar to the level of modularity of Learning Objects, could be achieved for the Virtual Learning Environment as well. Using the VWE Learning Object Taxonomy, this was accomplished by including the learning content and the Virtual Learning Environment into the same conceptual space. The comparison of the prototypes showed that the Web Service approach was preferred in favor of the Java-RMI approach. This was mainly due to platform neutrality and the use of the http-protocol. The study was supplemented by an analysis of the two approaches in relation to a third, REST-based approach.

INTRODUCTION

One of the greatest challenges for Technology Enhanced Learning (TEL) is to achieve flexibility in the use and implementation of learning technology. One of the most obvious needs for teachers is pedagogical flexibility; that is flexibility to choose pedagogical methods, to choose (or not to choose) functionality and "tools", as well as the flexibility given by the ability to change, and modify the Virtual Learning Environment (VLE) in a way that is responsive to how the pedagogical context develops – that is, the ability to modify

DOI: 10.4018/978-1-4666-0011-9.ch2.14

the VLE at run-time. This kind of *pedagogical flexibility* depends on *technical flexibility* in terms of *adaptability* and *adaptivity*.

The current practice of using "learning platforms" such as *Learning Management Systems* (LMS), does not handle flexibility very well. One problem is that the functionality is restricted to what is currently available in the used product, and even though many LMS implement standards for learning technology, they are still often considerably proprietary, and monolithic, which make them act as information silos. The use of standards is often limited, and commonly restricted to standards for digital learning content (DLC) rather than standards for system architecture, and infrastructure, such as API's, protocols and data formats. A consequence of this is a heterogeneous infrastructure, consisting of several isolated islands, and each new tool that is not a part of the LMS, easily becomes a new isolated island. This phenomenon is sometimes referred to as the "silo-effect" where each system owns and maintains its own data and functionality, and where no consideration is taken to the overall infrastructure, or to reciprocal interaction and reuse of information and services in a local or a global perspective. Th silo-effect is especially troublesome for LMSes since it makes the development of VLEs approach in an opposite direction than the rest of the web – compared to development trends such as represented by Web 2.0, where modularity and non-proprietary exchange of data, information and services are essential characteristics and carriers of the very concepts themselves. However, this situation is slowly changing as architectural standards that facilitate modularity and openness are slowly maturing and LMS vendors have started to adopt them. Such leading examples are the Sakai and the eFramework (see below), as well as specifications such as the IMS General Web Services (IMS, 2006), and the recent Common Cartridge specification from IMS, that supports packagings and exchange of data, learner information and learning content, as well as the notion of (still rudimentary) mash-ups in order to form simple composite applications the "Web 2.0 way" (IMS, 2008).

Paulsson (2008) argue for an approach emanating from a *learning architecture and a learning infrastructure* point of view, rather than focusing on LMS systems that are packaged as, more or less, monolithic products, which is currently a common practice. In order to achieve this there is a need for standardized architectural frameworks and reference models that support modularity and bring a holistic perspective to the learning infrastructure. Previous experiences show that modularity is an efficient approach to achieving enhanced flexibility, as well as other advantages, such as reusability, and better support for evolutionary development that, in the long run, lead to better stability and sustainability of the infrastructure. Component-Based Software Engineering is one area where modular approaches have been used for a long time and where such benefits are experienced, as described (See e.g. Williams, 2001; Szyperski, 2002; and Erl, 2007).

Modular approaches have been tested within TEL as well, and Learning Objects, that addresses modular learning content, is by far the most referred modular approach. Learning Objects are based on the idea of small, context independent, "chunks" of digital learning material that can be aggregated (to later be disaggregated again) to form larger units of learning content, sometimes referred to as *Learning Modules or Sharable Content Object as in SCORM (Thropp & Dodds, 2006)*, for use in a specific learning context. Learning Objects have been around for more than a decade, and is a well-established concept. However, in (Paulsson, 2008), (Paulsson & Naeve, 2006b), and (Paulsson & Naeve, 2006a), we argue that even though the concept is well-established, it is still not sufficiently defined in functional or in technical terms, to be useful in a way that give Learning Objects the characteristics that they are usually ascribed. In (Paulsson & Naeve, 2006a), we argue that Learning Object definitions must