Journal of Network and Computer Applications 32 (2009) 1092–1105

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Language engineering techniques for the development of e-learning applications

Iván Martínez-Ortiz, José-Luis Sierra *, Baltasar Fernández-Manjón, Alfredo Fernández-Valmayor

Dpto. Ingeniería del Software e Inteligencia Artificial. Fac. Informática, Universidad Complutense de Madrid, C/ Profesor José García Santesmases, s/n. 28040 Madrid, Spain

ARTICLE INFO

Article history:
Received 19 June 2008
Received in revised form
29 January 2009
Accepted 26 February 2009

Keywords:
E-learning applications
Language engineering
Domain-specific languages
Authoring
Model checking
Rapid prototyping

ABSTRACT

In this paper we propose the use of language engineering techniques to improve and systematize the development of e-learning applications. E-learning specifications usually rely on domain-specific languages that describe different aspects of such final e-learning applications. This fact makes it natural to adopt well-established language engineering principles during the construction of these applications. These principles promote the specification of the structure and the runtime behavior of the domain-specific languages as the central part of the development process. This specification can be used to drive different activities: rapid prototyping, provision of authoring notations and tools, automatic model checking of properties, importation/exportation from/to standards, and deployment of running applications. This language engineering concept also promotes active collaboration between instructors (the users of the languages) and developers (the designers and implementers) throughout the development process. In this paper we describe this language-driven approach to the construction of e-learning applications and we illustrate all its aspects using a learning flow sequencing language as a case study.

1. Introduction

A common practice in e-learning is the use of languages to describe the different aspects of a learning scenario (e.g. content, activities, participants, etc). IMS standardization efforts are good examples of this trend (Friesen, 2005). Indeed, many of these efforts result in suitable languages for the description of specific aspects of an e-learning application. Among them it is possible to find languages for packaging learning contents (i.e. the IMS Content Packaging specification) (IMS, 2004), for describing the different products involved in an assessment process (i.e. the IMS Question & Test Interoperability specification) (IMS, 2006), for describing the profile of a particular learner (i.e. the IMS Learner Information Package specification) (IMS, 2005), for sequencing the activities in a learning flow (i.e. the IMS Simple Sequencing specification) (IMS, 2003b), or even for characterizing the different teaching methods arising in heterogeneous learning situations (i.e. the IMS Learning Design specification) (IMS, 2003a).

While these standardization efforts stress the use of languages to solve interoperability issues (i.e. as vehicles that can be used by heterogeneous platforms to exchange information), in our works we have promoted a complementary philosophy: using suitable languages to describe applications that are generated by automatically processing these descriptions (Sierra et al., 2006b). This philosophy is shared by the approaches to software development based on domain-specific languages (Deursen et al., 2000; Mernik et al., 2005). According to these approaches software development is conceived as a *language engineering* process, where suitable domain-specific languages are specified, implemented and maintained for each application domain, and where software applications are described using these languages instead of general-purpose programming ones. These approaches are specially well suited to domains where having efficient mechanisms to norm the interaction between domain experts and developers is a must. E-learning is a paradigmatic example of these domains, since many times the cost of providing the contents and fine-tuning the final applications exceeds by several orders of magnitude the initial development cost of the software infrastructures where the applications will finally be deployed. The adoption of a language-driven approach in e-learning results in a more rational distribution of responsibilities among the participants in the development process. Instructors will be in charge of producing and maintaining the final applications, while developers act as language engineers responsible for formalizing and maintaining the languages used by the instructors. Developers are also in charge of the software infrastructure associated with such languages (including the generators used to produce the final running applications).

* Corresponding author. Tel.: +34 913947548; fax: +34 913947547.
E-mail addresses: imartinez@fdi.ucm.es (I. Martínez-Ortiz), jlsierra@fdi.ucm.es (J.-L. Sierra), balta@fdi.ucm.es (B. Fernández-Manjón), valmayor@fdi.ucm.es (A. Fernández-Valmayor).

1084-8045/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2009.02.005

We have successfully tested these language-driven principles in the development of several e-learning systems and applications (Fernández-Manjón and Fernández-Valmayor, 1997; Moreno-Ger et al., 2007; Sierra et al., 2006c, 2007b, 2008c), where we have tested the importance of using domain-specific languages to orchestrate the collaboration needed between instructors and developers. In these experiences we have also realized the feasibility of a complementary approach to the interoperability-oriented standardization efforts when adopting a language-driven process model. Instead of seeking universal solutions, we consider that it is also interesting to take a different approach by formalizing the languages already used by instructors in their specific learning domains. Since these languages are domain specific and part of the instructors' experience, they are more understandable and easier to use for the instructors than the more generic ones. It does not mean that standardization issues are ignored. Indeed, these issues can be subsequently addressed by appropriate importation/exportation modules. However, by adopting a language engineering perspective, where developers are constantly providing suitable linguistic support according to the particular needs of the instructors, it is possible to promote an instructor-centered development process, which results in instructors' deeper involvement in the production and maintenance of the e-learning applications. In this work we will mainly illustrate this *bottom-up* approach, similar to the experiences reported by Sierra et al. (2006a), although the techniques will also be largely applicable in a *top-down* manner, based on the use, specialization and adaptation of pre-existing languages, such as the one proposed by Moreno-Ger et al. (2006).

The present paper exposes the marriage between e-learning and language engineering. For this purpose, it describes and illustrates the different activities promoted by a language-driven approach in e-learning, and how these activities are organized around sound specifications of domain-specific languages. These specifications start with the characterization of abstract *information models* for the languages, which are described in both a conceptual and a formalized way. The structural formalization of the languages allows for the subsequent formalization of their runtime behaviors in the form of suitable *operational semantics*. The resulting specifications are used to drive many other language engineering activities. Indeed, these specifications can be readily used to build running prototypes of the languages in a straightforward way, which can be used to refine the structure and the semantics of these languages. They can also be used to provide notations that are more user-friendly for the instructors (e.g. a graphical notation), by specifying a mapping between these notations and the abstract information models. The resulting languages facilitate the automatic checking of properties, which results in better authoring support for instructors. Since the usual e-learning specifications are also language based, it is possible to connect these specifications using appropriate language translations. Finally, the resulting high-level designs can be easily deployed using the well-known model-view-controller (MVC) pattern (Krasner and Pope, 1988), which is typically used for organizing many modern web-based applications.

The structure of the paper is as follows. Section 2 motivates the language-driven approach by comparing it with conventional development models. In Section 3 we introduce a case study that will be used throughout the paper for illustrative purposes. In Section 4 we focus on the structural and behavioral specification of domain-specific languages. Section 5 is devoted to analyzing the different activities enabled by this language-driven approach. The paper finishes by presenting some conclusions and lines for future work (Section 6).

2. Language-driven development of e-learning applications compared to conventional development approaches

In a conventional development process model instructors are requirement providers, while developers act as application implementers. By using conventional requirement acquisition techniques, developers interview instructors to determine which resources (i.e. contents, support tools, etc.) must be incorporated in the application, as well as how the final users (e.g. instructors, learners, etc.) interact with these contents. For example, as result of the requirement acquisition process to develop an e-learning course, instructors determine, among others: the learning contents and tools needed, the structure of the course/lessons and the transitions between the different parts of a lesson or the whole course. With all this information, and using general-purpose programming languages and technologies (e.g. Java, XML, etc.), developers implement the application; for example, they can provide a web-based implementation by using a suitable framework for the development of web-based applications, such as Apache Struts (Goodwill and Hightower, 2003). Then, the developed application is evaluated by instructors (perhaps with the help of end-users), who eventually can discover some aspects to be improved in the contents or in the interactions. Thus, instructors propose modifications and/or improvements in the application to the developers, who produce an enhanced application, starting a new evaluation (e.g. instructors could include new content, modify existing ones, as well as modify the learning flow which governs the transitions between the different parts of the course). This iterative behavior, characterized by the production/modification of applications, finishes when a satisfactory application has been obtained but it needs to be started again when the application needs to be updated (Fig. 1a). In fact, the process model described resembles the *Analysis Design Development Implementation and Evaluation* (ADDIE) methodology extensively used for the development of e-learning

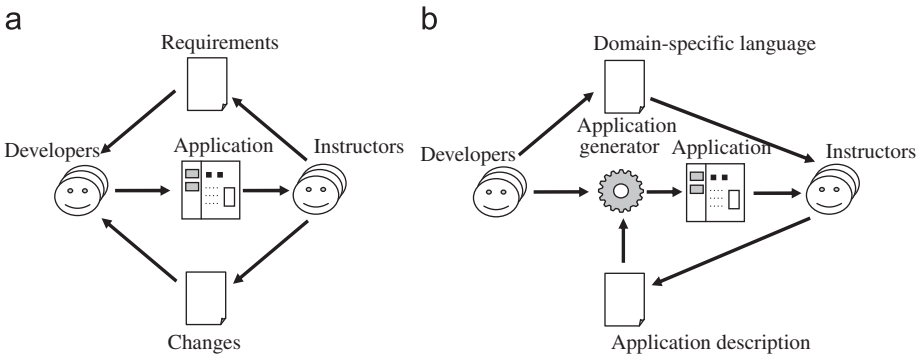

Fig. 1. Simplified views of (a) conventional development of e-Learning applications, and (b) language-driven development.