Language-Driven, Technology-Enhanced Instructional Systems Design

Iván Martínez-Ortiz, José-Luis Sierra, Baltasar Fernández-Manjón

Fac. Informática. Universidad Complutense de Madrid
C/ Prof. José García Santesmases s/n 28040 Madrid (Spain)
+34913947606

{imartinez, jlsierra, balta}@fdi.ucm.es

Abstract. In this paper we propose to extend the ADDIE (*Analysis – Design – Development – Implementation – Evaluation*) process for Instructional Systems Design (ISD) with a new *linguistic* layer. This layer allows developers to provide instructors with *domain-specific languages* to support and guide them through ISD. Instructors use the toolsets associated with these languages to produce technology-enhanced learning systems more effectively. We also describe how to put these ideas into practice by adopting modern model-driven software development processes together with the language engineering principles. This language engineering approach has been applied to <e-LD>, a highly flexible and extensible authoring tool for IMS Learning Design Units of Learning.

Keywords: Technology-Enhanced Instructional Systems Design, ADDIE, Software Language Engineering, IMS Learning Design, <e-LD>

1. Introduction

Instructional Systems Design (ISD) and the generic *Analysis – Design – Development – Implementation – Evaluation* ADDIE process were conceived as means of designing and developing learning systems, independently of whether these systems are technology-enhanced or not [2]. However, the introduction of a technological factor in the development process also introduces new issues that must be carefully addressed. One of the most important problems is the need to manage the active collaboration of *instructors* and *developers*. A way of addressing this collaboration is to use suitable *domain-specific languages* (DSLs) [10]. The application of DSLs results in a more rational distribution of roles: instructors use the languages to configure the technology-enhanced components, while developers provide the instructors with all the required machinery to make such a configuration possible.

In this paper we propose an extension of the generic ADDIE process model with a *linguistic* layer and illustrate this new process model using <e-LD> [6,7], an authoring tool for the production and reengineering of IMS Learning Design (IMS LD) Units of Learning (UoL) developed at Complutense University.

2. The Language-Driven ADDIE model

The *Language-Driven* ADDIE (LD-ADDIE) model is sketched in Fig. 1. This model is based on the revised ADDIE model proposed by the US Department of the Air Force (see [2]). It organizes the concepts and phases of the revised ADDIE model into five different layers. More precisely:

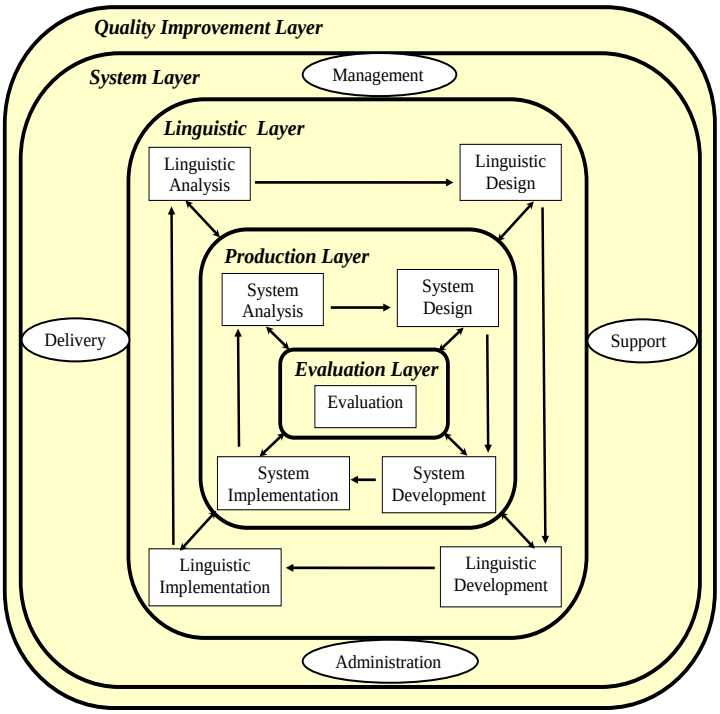

Fig. 1. The LD-ADDIE Model

- The *evaluation* layer includes activities centered on the continuous evaluation of the different aspects of the instructional system. It corresponds to the *evaluation* phase in the original ADDIE model.
- The *production* layer encompasses the systematic sequence of phases oriented to the production of the instructional system. It corresponds to the other four ADDIE phases (i.e., analysis, design, development and implementation).
- The *linguistic* layer contains phases for the systematic production of the domain-specific languages and the associated toolsets. Although these phases mirror the phases in the *production* layer, their purpose is very different: to develop the languages and tools used by instructors for the development of learning systems.
- The *system* layer contains the main functions of the learning system: *management*, *administration*, *support* and *delivery*.
- Finally, the *quality improvement* layer represents the mechanisms needed to carry out continuous quality improvement.

LD-ADDIE adds a new layer, the *linguistic* layer, to explicitly address the technological factor of technology-enhanced instructional systems. The aim of the

phases within this layer is to develop languages and tools. Also, they are mainly carried out by developers:

- During *linguistic analysis,* developers analyze the instructional domain addressed by the learning system and the vocabulary and terminology used by instructors. The goal is to determine the main terms and concepts in this domain, as well as the relationships between these concepts. This analysis can be carried out using standard *domain analysis* techniques, as understood in software and domain engineering [3].
- During *linguistic design*, developers specify the syntax and constraints of the domain-specific language, as well as its operational semantics. In modern software language engineering practice, the language usually will be equipped with several syntaxes [4]: an *abstract syntax*, in terms of which the operational semantics is defined, and one or several *concrete syntaxes*, oriented to facilitate the use of the language by instructors. All these syntaxes will be linked by suitable transformations. Operational semantics, in their turn, will specify how technology-enhanced components can actually be produced from utterances in the language. During this phase developers also conceive the tools associated with the language. Typical tools will be authoring tools based on suitable concrete syntaxes, as well as generators of the technology-enhanced instructional components.
- During *linguistic development*, developers build the toolset supporting the DSL. For this purpose, they can use well-established traditional techniques in the construction of language processors [1]. They can also adopt one of the emerging tendencies in software language engineering, based on model-driven software development concepts and the use of language workbenches [4].
- Finally, during *linguistic implementation,* the DSL and the associated toolsets are made available for instructors. These tools will be integrated into the final leaning system as part of the *support* function.

3. The Language-Driven ADDIE Model in Practice with <e-LD>

To illustrate the LD-ADDIE model, we use <e-LD>, an experimental and highly adaptable and extensible authoring tool for IMS LD UoL developed at Complutense University [6,7]. The tool supports three main functions:

- *Importation*. Using this function, instructors can load pre-existing IMS LD UoL. The function also produces useful information to understand the structure and behavior of each imported UoL: a *hypertextual view* (Fig. 2a), and a *dependency graph* with the representation of the dependencies among the design elements related to learning activity sequencing [8] (Fig. 2b).
- *Authoring*. Using this function, instructors can load pre-existing IMS LD UoL. This function lets instructors edit the description of a UoL. For this purpose, they use the visual notation detailed in [7] (Fig. 2c).
- *Exportation*. This function makes it possible to generate an IMS LD UoL automatically from an <e-LD> description. The core of the function is an automatic translation of flowcharts into rule-based systems [9].

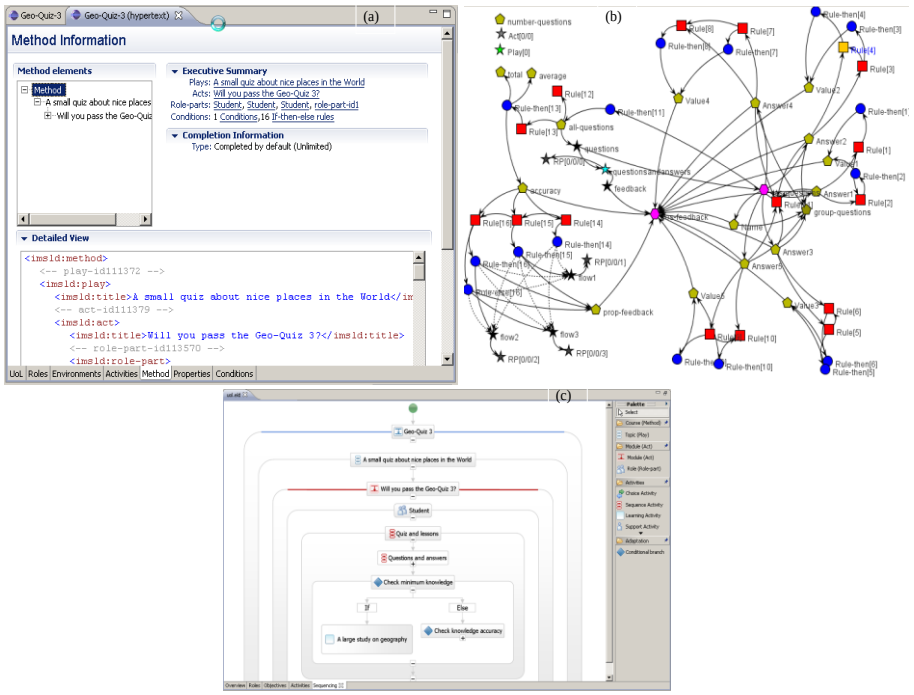

Fig. 2. (a) Hypertextual view of a UoL's method; (b) a dependency graph; (c) edition of a method in <e-LD>.

Since <e-LD> considers IMS LD UoL as essential parts of a learning system, it is possible to systematize the design of evaluation instruments in terms of the structure imposed by <e-LD> on such UoL (for example a satisfaction survey on a UoL can mirror the static structure of the UoL, say a method decomposed into several plays, each one integrating several acts, each one integrating several role-parts, etc.).

Also, <e-LD> plays a prominent role in the different production phases:

– During *system analysis*, instructors can find it useful to examine pre-existing UoL used in previous levels of instruction to determine the students' expected knowledge and capabilities, as well as to better determine the nature of the learning process and the more convenient performance exigencies.
– During *system design,* instructors can reuse pre-existing UoL in the instructional domain, importing them into the tool and modifying them in accordance with the target learning task. Also, instructors can use <e-LD> to author formalized plans of instruction for technology-enhanced components that effectively determines the instructional methods and strategies.
– During *system development*, <e-LD> provides a catalog to determine the different instructional resources and materials to be developed.
– Finally, during *system implementation*, instructors use <e-LD> to automatically generate standardized versions of the authored UoL encoded in IMS LD.

Regarding the linguistic layer, the development of <e-LD> follows the principles of modern software language engineering [4]. Indeed, the root of <e-LD> is a DSL developed using the language workbench provided by the Eclipse Modeling Project.

Thus, <e-LD> can be meaningfully conceived as the main product of an incarnation of the LD-ADDIE linguistic layer:

- As regards *linguistic analysis*, <e-LD> represents a cost-effective solution to the otherwise costly domain analysis processes. Indeed, <e-LD> reuses many of the conceptual structures of a pedagogically neutral language (IMS LD) with the hope of increasing the applicability of the solution while still maintaining a reasonable domain-specific nature.
- During *linguistic design*, the abstract syntax of the <e-LD> modeling language is characterized as a *metamodel* [4] that captures the main terms and concepts required to describe UoL in <e-LD>, as well as the relationships between these concepts, and the additional constraints affecting these elements. On the other hand, the concrete syntax corresponds to the aforementioned visual notation. These two syntaxes are related by an *abstract-to-concrete-syntax* mapping. Thus, by changing the concrete syntax model and this mapping, it is possible to tailor <e-LD> to the particular idiosyncratic requirements of each particular community of instructors. Finally, the operational semantics in <e-LD> are actually defined by the translation of flowchart-oriented specifications to rule-based ones used in the *exportation* function and described in [9].
- *Linguistic development* takes full advantage of the Eclipse Modeling Project. Indeed, the metamodels of <e-LD>'s abstract and concrete syntaxes are supported by EMF (the *Eclipse Modeling Framework*). Translation to IMS LD (carried out during *exportation*) is currently done as an *ad-hoc model-to-model* transformation; however, we are starting to refactor this process using the model-to-model transformation languages provided by the Eclipse Model to Model project. <e-LD> also takes full benefit of GMF (the *Graphical Modeling Framework* of Eclipse) to facilitate the development of the <e-LD> *authoring* function. Finally, the <e-LD> *importation* function is implemented as an XML processing component. We are currently refactoring it using XLOP (XML *Language Oriented Processing*) [12], an environment for the processing of XML documents with *attribute grammars* [11] also developed at Complutense University.
- Finally, during *linguistic implementation*, <e-LD> is deployed for the instructors as an Eclipse-based standalone authoring tool. Currently we are also working on integrating it with other IMS LD compliant platforms and tools, particularly IMS LD players.

Finally, following the guidelines encouraged by LD-ADDIE, <e-LD> is an integral part of the learning systems' *support* function. In addition, it is also subject to continuous improvement. The adoption of principles strongly rooted in software language engineering in its design and development facilitates this continuous improvement.

4. Conclusions and Future Work

In this paper we have described an extension of the ADDIE model for instructional systems design that highlights the collaboration between instructors and developers during the development of learning systems with significant technology-enhanced

components. For this purpose, the extension promotes the production of domain-specific languages and associated toolsets as support for instructors. The resulting model (LD-ADDIE) makes explicit a *linguistic* layer oriented to the systematic production of language-oriented assets. We have illustrated the model with <e-LD>, an authoring tool for IMS LD UoL. From a linguistic point of view, the development of <e-LD> takes advantage of the language workbenches provided by the Eclipse Modeling Framework.

We are currently applying the same principles to other language-driven e-Learning systems: <e-QTI>, a toolset for the authoring and deployment of QTI assessments [5], and <e-Tutor>, a system [13] for the description of Socratic tutorials. Finally, we plan to further experiment with the adaptation of (the concrete syntax of) <e-LD> to different communities of instructors in several instructional domains.

Acknowledgements

We wish to thank the projects TIN2005-08788-C04-01, TIN2007-68125-C02-01, Flexo-TSI-020301-2008-19, Santander/UCM PR34/07 – 15865 and CID-II-0511-A, as well as the UCM Research Group 921340.

References

1. Aho A.V, Lam M.S, Sethi R, Ullman J.D. Compilers: principles, techniques and tools (2nd Edition). Addison-Wesley. 2006.
2. Allen, CW. Overview and Evolution of the ADDIE Training System. Adv. in Dev. Human Res. 8(4), 430-441. 2006
3. Czarnecki, K. Generative Programming: Methods, tools and Applications. Addison-Wesley. 2000
4. Kleppe, A. Software Language Engineering: Creating Domain-Specific Languages Using Metamodels. Addison-Wesley. 2008
5. Martínez-Ortiz, I., Moreno-Ger, P., Sierra, J.L., Fernández-Manjón, B. <e-QTI>: a Reusable Assessment Engine. ICWL'06. 2006
6. Martínez-Ortiz, I., Sierra, J.L, Fernández-Valmayor, A., Fernández-Manjón, B. Language Engineering Techniques for the Development of E-Learning Applications. J. Network Comp. Appl., *in press* (DOI: 10.1016/j.jnca.2009). 2009
7. Martínez-Ortiz, I., Sierra, J.L., Fernández-Manjón, B. Authoring and Reengineering of IMS Learning Design Units of Learning. IEEE Trans. on Learning Tech., 27 Mar. 2009. <http://doi.ieeecomputersociety.org/10.1109/TLT.2009.14>. 2009
8. Martínez-Ortiz, I., Sierra, J.L., Fernández-Manjón, B. Enhancing IMS LD Units of Learning Comprehension. ICIW'09. 2009
9. Martínez-Ortiz, I., Sierra, J.L., Fernández-Manjón, B. Translating e-learning Flow-Oriented Activity Sequencing Descriptions into Rule-based Designs. ITNG'09. 2009
10. Mernik M, Heering J, Sloane AM. When and how to Develop Domain-Specific Languages. ACM Comp. Surveys. 37(4),316-344. 2005
11. Paakki J. Attribute Grammar Paradigms – A High-Level Methodology in Language Implementation. ACM Comp. Surv. 27(2), 196-255. 1995
12. Sarasa. A., Sierra, J.L., Fernández-Valmayor, A. XML Language-Oriented Processing with XLOP. WAMIS'09. 2009
13. Sierra, J.L., Fernández-Valmayor, A., Fernández-Manjón, B. From Documents to Applications Using Markup Languages. IEEE Software 25(2), 68-76. 2008