Chapter VI
A Step towards a Pattern Language for E-Learning Systems

Andreas Harrer
University of Duisburg-Essen, Germany

Alke Martens
University of Rostock, Germany

ABSTRACT

Computer-based teaching and training systems in general, and Intelligent Tutoring Systems (ITSs) in particular, are usually based on similar fundamental structures. In contrast to this, analyses of the state of the art of teaching and training system development have revealed that software engineering techniques are seldom used for realizing those systems. In the last years, some approaches tried to change this: pattern mining took place; methods covering the specifics of ITS project development have been deployed. These approaches usually focus on a specific system type or on a certain application domain, thus re-usability is often not possible. What is missing is a combination of different approaches in a pattern language or a pattern catalogue for ITS. The purpose of such a pattern catalogue is to provide pattern for different types of software and to support the software development starting from design and ending with the implementation. A step towards a pattern language for ITS is described in this paper.

INTRODUCTION

The term e-learning covers a set of quite heterogeneous types of systems, for example, tele-teaching, CD-Rom, and various computer-based teaching and training systems (Kaplan-Leiserson, 2002). The type of e-learning system, with which the analysis described in this chapter is started, is the

intelligent teaching and training system (ITTS), which is also often called intelligent tutoring system (ITS).

Intelligent tutoring systems can look back on a comparably long history. They are usually based on similar fundamental structures, which are called the ITS architecture. This architecture consists of the following components: expert module (encapsulating expert and pedagogical knowledge), learner module, user interface module, and steering module. This architecture has been described by several researchers, for exmple, by Clancey (1984), Lelouche (1999), and Martens (2003). The ITS architecture can be seen as a pattern in the broad sense (Devedzic, 2001). But whereas a pattern should support comparable and similar realizations of software, the pure architecture description does not provide guidance in software development. Thus, in ITS design a situation has developed, where the component of different systems provide a quite homogeneous naming (with only slight variations, e.g., user model instead of learner model) but are in most cases based on a completely heterogeneous and non-interchangeable realization. Moreover, the interpretation of each component's role and functionality varies a lot (Johnson et al., 2000; Martens, 2003). What makes the situation even more confused is the insight, the term 'component' is seldom used in the software engineering sense (Szyperski, 2002), but more often in the common language sense as equivalent of 'part of' or 'constituent.' One result of this situation is that—from the perspective of computer science—ITSs are hardly comparable.

In recent years, several approaches have been made toward establishing uniformity in the development of teaching and training systems. Two main directions of research can be distinguished:

- Development of standards (or similar approaches, which intent to predefine terms, e.g., nomenclatures or metadata description).
- Development of software engineering methods.

Examples for standards and similar approaches are standardized architectures like the learning technology system architecture (LTSA) (see http://www.edutool.com/ltsa; Farance & Tonkel, 2001), approaches for learning project management like Essen learning model (ELM) (Pawlowski, 2000), and Extensible Markup Language (XML) descriptions for teaching and training content like the Dublin core metadata (see http://dublincore.org; DCMI, 2002). The main advantage of these approaches is the predefinition of terms, which is a step towards exchangeability. Main drawback is that the contribution remains on the level of terms. Standards and similar approaches usually lack a specific description of how to realize and implement the system parts.

Examples for an explicit focus on software engineering methods in ITS development are not so widespread (e.g., Illmann et al., 2000) and systematic theoretical approaches can rarely be found (e.g., Devedzic, 1999; Devedizc & Harrer, 2002; Harrer, 2003). Using methods of software engineering in teaching and training systems is based upon the idea that software development is mainly an engineering science, not an art—although the latter perspective usually describes how software development is done in several different branches of computer science, and also in the area of ITS. To organize work according to the engineering perspective has some consequences: the development of software is based on techniques, which are clear, definite, and traceable, and which explicate different aspects of the development process and of the result (i.e., the software).

Taking up methods of software engineering and clear definition of concepts, this chapter describes a stepwise approach to develop a pattern language for e-learning systems of the ITS