Software Engineering in E-Learning Systems

Alke Martens
University of Rostock, Germany

Andreas Harrer
University of Duisburg-Essen, Germany

INTRODUCTION

Regarding the role of software engineering in the development of different types of e-learning systems, a traditional situation is contrasted with a modern state of the art. Traditionally, these systems, especially the intelligent tutoring systems (ITS) or artificial intelligence in education systems (AIED), are developed as research projects (Harrer & Martens, 2006). This means a comparably small group of people is involved in system development; the systems are developed with a research focus and not with respect to reusability, maintenance, robustness, or extensibility. The systems usually are not sold or used over long periods of time. In recent years, e-learning systems have reached maturity. Several e-learning systems are available as software products. They have left the stage of pure research and can now be found in relation with the buzzwords "everyday and lifelong learning." In particular generic e-learning systems, in contrast to the more research oriented ITS, are nowadays often developed based on software engineering techniques.

The lack of usage of software engineering in research oriented e-learning systems has led to a situation where the resulting systems can hardly be compared, and communication about the existing systems is difficult. Even if a common agreement about the underlying system architecture exists, for example, the classical ITS architecture (Clancey, 1984; Martens, 2003) or the learning technology system architecture (LTSA) (Farance & Tonkel, 2001), system components usually can not be reused (for an analysis see e.g., Martens, 2004). E-learning projects often reinvent the wheel. The extension of existing research oriented e-learning software and its further development (potentially beyond pure research) seldom takes place; surprisingly, as one characteristic of software is its malleability. Why it is important to use software engineering in e-learning system development—especially in research oriented e-learning—and which methods can be used will be highlighted in this article.

BACKGROUND

In the early 1960s, the so-called software crisis has led to a new field of research, which has been named *software engineering* (Ghezzi, Jazayeri, & Mandrioli, 2003). Up to this date, software development has had a strong focus on programming as a personal activity. Few people participated in the development process. Programming was (and sometimes still is) perceived to be an art (Hunt & Thomas, 2001, 2006; Knuth, 1974) rather than a craft (Reynolds, 1981).

Since then the growing complexity of software systems has led to a situation where (changing) teams of developers participate over a long period of time on the development process. Nowadays, software itself is usually multiversioned, that is, it is changed during lifetime. Communication between a lot of different people about software and software development has to take place. Component oriented development, which allows different developers to work on one single project, and a clear development strategy are required. Most software systems are constructed for long-term use. Their development has to provide for reusability of components, on maintenance, robustness, reliability, correctness, and extensibility. Thus, one claim from the perspective of software engineering is to carefully design and to "engineer" software. Engineering should take place regarding the development process of the software and also regarding the resulting product. Software engineering comprises rigor and formality in the development process, the anticipation of change and extensibility regarding the resulting product, as well as modularity and abstraction at different levels of development.

E-learning can also look back on a comparably long history. The term was coined in the 1970s and has gained popularity in the last years. E-learning is related to several different types of systems. As the e-learning Glossary of Terms (Learn.com, 2007) summarizes, the term covers:

> *A wide set of applications and processes, such as Web-based learning, computer-based learning, virtual classrooms, and digital collaboration. It includes the delivery of content via Internet, intranet/extranet (LAN/WAN), audio- and videotape, satellite broadcast, interactive TV, CD-ROM, and more.*

In this sense, e-learning can be seen as every form of *electronically supported learning*. Regarding computer based learning in particular, several types of systems are embraced by the term e-learning, for example, computer based training (CBT), adaptive hypermedia (AH), intelligent tutoring systems (ITS), and game-based learning systems, to name but a few. There are some modern system types which are also part of e-learning, for example Web-portals (PostNuke, 2007), learning management systems (LMS) (Avgeriou, Papasalouros, Retalis, & Skordalakis, 2003), and platforms for structured discussion forums like the future learning environment (FLE3) (Muukkonen, Hakkarainen, & Lakkala, 1999). These approaches rely on software engineering techniques, that is, modularization, usage of patterns, and extensibility. These systems can more or less easily be extended by other developers, as it has been done for example, by Dolonen, Chen, and Morch (2003). In contrast to this, research oriented systems, like AIED systems and ITS, are seldom developed based on software engineering. Surprisingly, as these systems often have an inherent complexity. ITS and AIED systems combine different fields of research, like artificial intelligence and cognitive science. Thus, research projects consist of people with different research background. ITS and AIED systems are based on similar architectures (Martens, 2003). Usually they are developed according to a certain pedagogical or learning psychological theory. Unfortunately, in most cases neither system components nor underlying system concepts can be reused. Existing systems are often not malleable. Most of these systems lack a detailed description, which makes communication difficult. Due to these facts, the resulting systems are hardly comparable as insights are often not portable. The combination of similar underlying architectures, a heterogeneous research background in project partners, and a complex system calls for the usage of software engineering techniques in ITS and AIED system development.

Naturally, when it comes to system development, no silver bullet exists, which could cover the broad field of e-learning applications. Thus, in the following, sometimes the type of system, for which the sketched software engineering approach has been invented, is explicitly named.

METHODS OF SOFTWARE ENGINEERING IN E-LEARNING

In software engineering, the process of software development as well as the resulting product should be "engineered." Engineering as an activity comprises a clear definition of the development process and a distinct vision of the resulting product. There are several methods and techniques, which can be applied in the design and development of e-learning software. Some of them are named in the following.

At the *project level* interdisciplinary communication between experts from computer sciences, experts from learning sciences, and experts from the training application domain is required and calls for project management approaches. At the project level, a careful design of the *software production process* (e.g., the waterfall model or the spiral model) (Ghezzi et al., 2003) can facilitate communication. Each expert has a field of responsibility and certain concerns regarding the resulting product. For example an expert from the application domain is responsible for the development of the expert knowledge module of an ITS. But the expert is not necessarily interested in the implementation details, like data storage and data retrieval. Also, a computer science expert, who implements the expert knowledge module, does not need to become an expert of the application domain. Given an appropriate level of abstraction can help the different expert groups to organize responsibilities and can facilitate communication about the system to develop. Software design would separate the application domain oriented specification (e.g., domain knowledge) and technical aspects of implementation. One approach in this direction has been made by the Essen learning model (ELM) (Pawlowski, 2000). ELM is a *software*