Abstract

Instructional software can be defined as any computer program used for instruction. As a means to optimise the development process of instructional software, different authors have defined a range of methods based on Software Engineering. However, the use of a method does not guarantee the validity and quality of the product developed; hence, a series of Software Process Improvement (SPI) approaches aimed to improve and guarantee the development process that have arisen.

The purpose of this paper is to review the aforementioned approaches and to apply them to the analysis phase of the instructional software development process. The analysis phase aims to define the conditions and needs of the project in order to solve uncertainties in early stages of the process.

1. Introduction

The development of instructional software is a complex task that encompasses diverse disciplines. Given the difficulties attached to this process, a range of authors have focused on the design of tools [7], techniques [1,6,12] and methods [8,11] that can facilitate and improve the development of computer-based learning systems. All these approaches share one characteristic: the application of Software Engineering in the field of Instructional Design and Technology (IDT) as a means to improve the development of instructional software.

Following this trend, and given the efficiency of the outcomes, it seems convenient to adapt existing process improvement models from Software Engineering to the development of computer-based learning systems. Software development process improvement provides a means for the qualitative evaluation of processes without the need to follow a specific methodology [4].

The purpose of this paper is to make an examination of these process improvements and to apply those findings to one of the crucial phases of the development process: the analysis phase. The final objective of our research will be therefore to have an analysis method for learning systems that, based on these models, would guarantee the quality of the analysis. This analysis improvement can facilitate the automation of consecutive stages of the process.

The rest of the paper is organized as follows: Section 2 briefly examines the applicability of SPI models to IDT. Section 3 describes an analysis method that embeds such knowledge. Final conclusions and a set of recommendations for future research are compiled in the last section.

2. A Capability Learning Analysis Method

The analysis stage is an essential phase of the software development process [9]. The importance of the analysis task increases with the size and complexity of the system: requirements should be determined in early stages to reduce their unexpected impacts. This section aims to present a compilation of the learning points acquired from the review of SPI frameworks.

2.1 Software Process Improvement

Software Process Improvement (SPI) is defined as:

“A deliberate, planned methodology following standardized documentation practices to capture the activities, methods, practices, and transformations that people use to develop software and the associated products.” [10]

According to this definition, SPI approaches are oriented to improve products and processes that take into account knowledge and general experience from previous software development projects.
Based on this conception, SPI frameworks for general application have been devised. Because of their primary significance, the following models should be mentioned:

- **Capability Maturity Model Integration (CMMI)** is an SPI approach that provides organizations with the essential elements of effective processes. It can be used to guide process improvement across a development phase, a project, or an entire organization [4].
- The **Software Process Improvement and Capability Determination (SPICE)** project, also known as ISO/IEC 15504, is a SPI framework for the assessment of software processes [5]. It defines a two-dimensional reference model for describing processes, activities and the process capabilities used in an assessment model.

### 2.2. Applicability of the Process Improvement for the Learning Analysis Method

In our research area, it seems reasonable to ask whether the concept of process improvement is relevant for the field of IDT. According to different literature references [3], our answer will be affirmative; affirmation that we can authenticate with the following assertions:

- CMM and SPICE can be applied to any software development process. They do not address expertise in any particular domain.
- CMM and SPICE offer a generic improvement framework. They provide a means for the software development improvement without the need to follow a specific methodology.

Another question that needs to be answered is related to the application of the SPI frameworks to the purpose of this research: the analysis phase. As already indicated in the previous section, CMM and SPICE can be applied to a particular stage of the process.

### 2.3. Lessons from SPI models

There are a series of reviewed points common to any software development process. These gains are:

- The quality of a system is highly influenced by the quality of the process used to develop it.
- The SPI frameworks establish a common arrangement for discussing and assessing software process and methods.

The next reviewed points have been taken from the review on the engineering and support process areas. These subjects focus on the process capability. The result of this revision is translated as follows:

- Elicitation. In the case of learning systems, the problem must be analyzed from diverse points of view. These perspectives reflect the customer requirements, the stakeholder requirements and the development product requirements.
- Specification. The analysis phase aims to gather and study the requirements of the system. As a result, a mere compilation or a narrative presentation of the system characteristics is not sufficient.
- Management. A managed learning analysis method must establish and maintain the integrity of analysis products in order to avoid inconsistencies and to manage changes.
- Validation. The activity of determining whether or not the analysis artifacts meet all of the input requirements for the analysis phase.
- Verification. Verification refers to the activity of evaluating analysis to ensure that it is free from failures. A failure is defined as an incorrect product behavior.
- Measurement. Measuring what analysis does and then using those measures to control and improve analysis activities should be an important practice.
- Alternatives. This practice is needed where stakeholders face a number of options, and no optimal choice is obvious.
- Discovering defects. An optimized analysis method should have techniques to discover and identify defects in products and processes, and then remove them as appropriate.

Bearing in mind these indications would allow us to define a process for an effective and optimal analysis.

### 3. Learning Analysis

Learning Analysis (LA) is conceived as the collection of activities and artifacts that enable us to improve the analysis phase in the learning materials development process. Such techniques, activities and artifacts, are based upon lessons from SPI models.

#### 3.1. Engineering Principles

The method suggests the following ideas:

- Multi-Dimensional approach. LA method uses Aspect-Oriented and Model-Driven techniques to facilitate the development of complex learning systems. These approaches will allow solving the existing overlap between the different views of the problem [2]. Related lessons: elicitation and specification.
• Heuristics. Heuristics are rules which guide the requirements specification towards a high probability of success. Related lessons: specification, management and verification.

• Analysis model. The analysis model is used to capture the purpose of the system by describing its behaviour. This behaviour will be expressed in terms of objectives and goals. Related lessons: specification, validation and verification.

3.2. Support activities

These are activities from LA that provide services to the rest of the elements of the method.

• Configuration management. It establishes and maintains the integrity of analysis products by identifying the configuration items, controlling changes to these items, creating baselines and performing configuration audits. Related lessons: management and measurement.

• Traceability. LA method provides bidirectional traceability in order to identify inconsistencies. Related lessons: management and measurement.

• Success indicators. These indicators are used to determine if the goals have been met. These indicators assess the requirements consistency and the method efficiency. Related lessons: measurement.

From the learning points presented above, it can be concluded that the LA method does not contemplate the automatic proposal of alternatives or the identification of faults and defects.

4. Conclusions and further work

The application of Software Engineering to instructive design has resulted to be an appropriate working strategy to optimise the development of learning systems. According to this idea, we have reviewed different process-improvement models and applied their learning points to the analysis stage.

The review of process-improvement models has shown some of the characteristics that should be considered in an optimized analysis method in order to guarantee the quality and validity of the process. In addition, the Learning Analysis has been presented as an analysis method for learning systems which guarantees their validity and adapts to the demands of process improvement models.

Future work should be oriented to the inclusion of all the learning points gained in Learning Analysis and to the application of process improvement models to other stages of the development process.

5. Acknowledgements

This work is supported by the Moduweb project (TIN2006-09678) funded by the Spanish Ministry of Science and Education (MEC).

6. References


