Chapter 15
ECSE:
A Pseudo-SDLC Game for Software Engineering Class

Sakgasit Ramingwong
Chiang Mai University, Thailand

Lachana Ramingwong
Chiang Mai University, Thailand

ABSTRACT

Software development is uniquely different especially when compared to other engineering processes. The abstractness of software products has a major influence on the entire software development life cycle, which results in a number of uniquely important challenges. This chapter describes and discusses Engineering Construction for Software Engineers (ECSE), an effective workshop that helps software engineering students to understand some of these critical issues within a short period of time. In this workshop, the students are required to develop a pseudo-software product from scratch. They could learn about unique characteristics and risks of software development life cycle as well as other distinctive phenomenon through the activities. The workshop can still be easily followed by students who are not familiar with certain software development processes such as coding or testing.

INTRODUCTION

The intangibility of software makes it a highly unique product (Project Management Institute, 2008). Undeniably, the development of software has a number of different traits compared to other engineering products. In a software development project, several issues, such as potential frequent changes of requirements, low product visibility, inappropriate development models, and the need for customer involvement, are critical (McConnell, 1997; Schmidt, Lyytinen, Keil, & Cule, 2001; Tiwana & Keil, 2004). Although these challenges can be addressed in traditional lectures, it is highly unlikely that the students can actually follow and understand their practical seriousness. For example, a lecturer can describe how difficult and costly it is if a major change surfaces during the

DOI: 10.4018/978-1-4666-5800-4.ch015

last stages of development. Yet, it is not easy to imitate other associated issues such as conflicts between stakeholders, frustration and the importance of problem-solving and negotiation skills.

Indeed, the most effective method to teach software engineering is to have the students learn through an actual hands-on project. Yet, regardless of whether a traditional or agile model is chosen, the implementation usually takes days before the results can be clearly seen. Furthermore, hands-on project could become considerably less effective if the group is bigger. Additionally, since general hands-on projects mostly focus on coding, students who have less relevant skills usually feel uncomfortable and subsequently fade away from the workshop. Indeed, this could be one of the least desirable outcomes from the class.

A number of researchers have attempted to implement games in their classes in order to overcome such challenges (Caulfield, Xia, Veal, & Maj, 2011). These games can be roughly divided into two groups, i.e. traditional games and computer-based games. Traditional games involve activities in which the students can participate by using convenient physical tools such as paper, scissors, boards, cards and dice. On the other hand, computer based games uncomplicatedly refers to games which the students need to play via a computer application. Some of these games can be played in groups while others support only a single player mode. Moreover, the settings and requirements of these games generally vary. Many of these software engineering games are flexible and can be further tailored to match class objectives.

In-class competition can be an important factor to increase the workshop's effectiveness (Hainey, 2009). With this factor included, the students are more likely to put more attention to the class. They also tend to perform their actions more seriously and carefully.

This chapter introduces *Engineering Construction for Software Engineers (ECSE)*, a game that attempts to teach and simulate a complete concept of software development life cycle in only two and a half hours. Instead of developing real software, the students are instructed to build a model house from corrugated plastic board. During the activity, the participants can learn basic knowledge of software engineering and the Software Development Life Cycle (SDLC). Although software development skills are not required, it can greatly benefit the team. ECSE also implements a currency and resource management system in order to increase fun and competitive factors. The students are required to plan and appropriately allocate their budget. There is no limitation on implementation strategies and approaches. The winner is, undoubtedly, the group which makes the most profit from the entire process.

BACKGROUND

The software development life cycle (SDLC) varies based on the nature of software developers and software organizations. The classic SDLC consists of five major phases i.e. requirements, design, construction, testing, and maintenance. This entire cycle can be further tailored based on business needs. Common modifications of the SDLC include the expanding, grouping, and revolving of existing phases as well as adding specific activities such as initiation, prototyping, and retrospection.

On the other hand, agile software development models have their own manifesto. They place emphasis on frequent working product delivery and do not follow the classic SDLC sequence. Agile practitioners value changes, interactions, and collaboration above plans, tools, and contracts (Beck et al., 2001). Yet, agile and traditional developments inevitably share similar activities. Indeed, in the same way as the implementation of the traditional SDLC, agile processes can also be tailored based on business needs and development environments.