Construction of a Software Development Model for Managing Final Year Projects in Information Technology Programmes

https://doi.org/10.3991/ijet.v15i21.15401

Abdelrahman Osman Elfaki (✉), Zaid Bassfar
University of Tabuk, Tabuk, Saudi Arabia
a.elfaki@ut.edu.sa

Abstract—The final year project (FYP) is considered a capstone in information technology (IT) programmes and involves the development of a software product. Currently, students are using the traditional software development life cycle approach to manage their FYPs. However, this approach can cause many difficulties. This paper proposes an alternative software development model for managing the FYP in an IT and IT-related degree programmes of study. First, a benchmark exercise was undertaken to compare the software developed for business purposes with that developed for educational purposes, which took into account the ten project management knowledge areas. The result of this exercise indicated that the differences that exist between business and educational software necessitate the development of a new software development model that is specifically tailored to the development of educational software. Therefore, capability maturity model integration (CMMI) was modified to generate a new version of CMMI – named educational CMMI – which could be used to evaluate educational software projects and detailed mathematical descriptions of the proposed model were composed. As conclusions, the proposed model was then assessed by students' results and by questionnaire feedback, the results of which showed that the proposed model was both useful and applicable for its intended target users and context.

Keywords—Capstone Project, Software Developing Methodology, Project Evaluation

1 Introduction and Motivation

Software is about developing not manufacturing [1]. This is a very famous statement in the software engineering community, which reflects the special nature of software. Moreover, it is recognized that developing software requires art skills in addition to scientific skills. This special nature of software means that software engineering differs from other engineering fields in terms of the required inputs and tools. Hence, traditional project management methodologies are not suitable for software projects and there is therefore a need for dedicated software project methodologies. These software project methodologies are also known as software development meth-

odologies, system development methodologies, the software development life cycle (SDLC) or the software development process.

A software development methodology can be defined as "a splitting of software development work into distinct phases (or stages) containing activities with the intent of better planning and management" [2]. Currently, the common and traditional software development methodologies are waterfall, prototyping, iterative and incremental development, spiral development, rapid application development and extreme programming methodologies as well as various types of agile methodology [3]. The main aim of all of these software development methodologies is to produce high-quality software products. These methodologies comprise a set of processes that are designed to work rigidly and sequentially. This strong hierarchical approach is intended to guarantee a high level of control over software projects [4]. Software developers seek to have high control over their projects in order to produce high-quality software products. Meanwhile, from the business perspective, high-quality software is important because it can best satisfy customer needs.

On the other hand, there is another type of software development, which is the development of software for educational purposes. University students enrolled on programmes of study in IT and computer science are required to take a compulsory subject called the graduate project [5]. To fulfill the requirements for passing this undergraduate project, students must develop a software program. Hence, the purpose of this software is completely different from that of other traditional software. In short, the intention of the undergraduate software project is to achieve pedagogical and educational outcomes.

Based on the above discussion, there are two different targets when developing software:

i. Commercial or business
ii. Educational purposes

However, despite these two completely different targets, the project development methodologies are the same. The lack of a specific software project methodology for educational purposes causes many difficulties for students, such as not having access to the correct guidance that should be provided by a development methodology. Therefore, in section two of this paper, the problem of this lack of a specific software development methodology for educational purposes is discussed in depth.

In this paper, for the reader's convenience, we have labelled software that has been developed for business purposes as 'business software' and software that has been developed with an educational purpose in mind as 'educational software'. We have also labelled the graduation project or software capstone project which is a compulsory component of an undergraduate (bachelor's) degree in computer science, software engineering and IT as the final year project (FYP).

In this paper, there is a tailored methodology that has been followed to complete the requested work. In other words, we have designed our own methodology to develop and prove the proposed model. The tailored methodology consists of:

- Conducting a benchmarking exercise based on ten project management knowledge areas, to compare the business software against educational software. The output of this step has emphasized the necessity of developing new educational software development model.
- Adjusting CMMI to generate a new version of CMMI which is educational CMMI. The output of this step is the proposed model.
- Conducting experiments to evaluate the proposed model. The output of this is direct assessment of the proposed model.
- Collecting and analyzing students' feedback regarding the proposed model. The output of this is indirect assessment of the proposed model.

The remainder of this paper is structured as follows: Section two presents a comparison of business and educational software, which is based on the ten project management knowledge areas. This section highlights the critical need for a new software project methodology that is tailored specifically for software projects for educational purposes. Next, section three reviews the related works and describes the research gap. Then, section four provides a detailed description of the proposed model for educational software development. After that, section five presents the implementation of the proposed model. Finally, section six discusses the outcomes of this study and draws some conclusions.

2 Comparison of Business Software and Educational Software

According to [6-10], a number of difficulties are encountered when using standard software development methodologies in the field of education. These include difficulty in applying self-directed learning, difficulty in finding a project task that engages all participants equally and difficulty in preparing some learning tasks (e.g., interpersonal and structural) for the project task. From this finding by [6], it can be concluded that there is a critical need for a new software model for managing the FYP in IT and IT-related programmes of study.

To highlight the differences between business and educational software, we compared these two types of software according to the ten project management knowledge areas defined by the Project Management Institution [11]. According to [12], any development of software could be considered as a project. Therefore, these project management knowledge areas were selected as the benchmark criteria for this comparison. In Table 1 below, each knowledge area in [11] is described from the business software and educational software perspectives. These descriptions provide suitable measurable conceptualizations by which to make the comparison.

Table 1. Differences between Business Software and Educational Software

#	Comparison criteria	Business software	Educational software
1	Project Integration Management: Purpose	To satisfy stakeholders' requirements	To satisfy learning objectives
2	Project Scope Management: Software Requirement Specification (SRS)	The SRS should be realistic and extracted from real stakeholders' requirements and should be approved by stakeholders	The SRS represents a sample of real requirements and should be approved by the academic supervisor
3	Project Time Management: Time commitment	Has acceptable flexibility	Does not have flexibility
4	Project Cost Management: Cost commitment	Restricted in terms of the planned cost. Receives the highest level of attention	Flexible in terms of the planned cost. Does not receive the highest level of attention
5	Project Quality Management: Quality commitment	Measured according to business quality metrics	Measured according to academic quality metrics
6	Project Human Resources Management: Evaluation of the role of each member in the team	Is not considered as a success factor in a project evaluation	Is considered as an important success factor in a project evaluation
7	Project Communications Management: Commutation plan	Related to satisfying specific milestones	Related to the academic plan
8	Project Risk Management: Testing	Is a crucial and significance factor in acceptance of the software	Is not a crucial factor
9	Project Procurement Management: Procurement of software and hardware	Critical and needs careful planning	Done mainly by the academic supervisor
10	Project Stakeholder Management: Documentation	Process documentation for internal usage, or user manual	An academic document in the form of a thesis

Each of the above comparison criteria is discussed in detail below:

- **Purpose**: This criterion refers to the final aim of the developed software. Business software is aimed at satisfying stakeholders' requirements, the achievement of which should be reflected positively in financial profit to the company. On the other hand, educational software is targeted at three issues: providing students with technical knowledge, providing students with practical experience and evaluating students' knowledge. Thus, business and educational software have completely different purposes.
- **Software Requirement Specification (SRS)**: In business software development, the SRS must be realistic and should therefore be extracted from stakeholders' actual requirements and needs. This element of the development process is regarded as having the highest importance. In the case of educational software, the content of the SRS mimics the problem domain rather than truly reflecting an actual case. Although it is important to try to present a realistic SRS, representation of the problem is usually adequate. However, oftentimes, students waste much of their limited time in focusing on this stage of the process.
- **Time commitment**: Although the amount of time needed to develop a software solution for a real client should be mentioned in the contract for any business software, there is still a degree of flexibility in the overall time frame due to the im-

portance of the finished product achieving stakeholder satisfaction. On the other hand, educational software is restricted by a definitive time frame because the developed software must be submitted for evaluation at a predefined point in time in the programme of study – regardless of the level of progress achieved – in order that it can be assessed and graded by the examiner/examining body for its contribution to the overall degree. Hence, it is clear that business software has acceptable flexibility in regards to time commitment, whereas educational software does not.

- **Cost commitment**: The main target of developing business software is to achieve a profit. Therefore, the costs involved in development are necessarily restricted. Indeed, cost is the most important consideration in business software development. In contrast, educational software is targeting academic benefits and is therefore very flexible in respect of the cost.
- **Quality commitment**: The quality of business software is measured according to standard business quality metrics, which are entirely different from the metrics used to assess FYP software.
- **Evaluation of the role of each member in the team**: This criterion refers to measuring the individual contribution of each member of the development team toward the total effort exerted in the software development process. In the case of business software, the distribution of roles between the team members is an internal issue that is considered only internally within the project management. The distribution of roles among the teams in business software development never appears to be discussed or evaluated as a software project success factor. On the other hand, in the case of educational software, the distribution of the roles among the members of the student team and the evaluation of the contribution of each member are both very important factors in the project evaluation process.
- **Commutation plan**: This criterion relates to moving from one phase of the development process to the next phase: In the context of business software development, the moving from one phase to another is controlled by some specific predetermined milestones. Meanwhile, in the case of educational software, moving from one phase to another is related to the academic plan and semester timetabling. The discussion time that has been defined by the academic plan is a sharp time for ending the educational software. Nevertheless, in business software, the development process cannot be finished until the customer accepts the software.
- **Testing**: In business software, testing is a crucial and significant issue. The final acceptance of a software solution mainly depends on the user acceptance test in which all the stakeholders' requirements should be considered and validated. In the educational software development process, testing should be implemented but it is not a condition for acceptance. Rather, testing is considered as an academic issue.
- **Procurement of software and hardware**: In the business software domain, procurement of original software and high-quality hardware is vital for a project's success. On the other hand, in the educational software domain, free and open source software is preferable due to cost constraints. In fact, the majority of students avoid purchasing expensive hardware and prefer instead to depend on simulators.

- **Documentation**: In the case of business software, there are two types of documentation: 1) documentation for internal usage inside the developer company, which is confidential and 2) documentation for the users, or a user manual, which describes how to use the software product. In the case of educational software, the only documentation is a scientific text in form of a thesis which is aimed at fulfilling academic requirements.

The above discussion clearly indicates that there are key differences between business and educational software. Hence it is obvious that the current software development methodologies are not sufficient to provide students with the necessary guidance on how to develop educational software and that the traditional methodologies fall short in some areas, particularly:

- Evaluating educational software based on educational objectives and functional requirements
- Evaluating teamwork by evaluating the role of each member in a team

Thus, it can be concluded that there is a pressing need for a new software development methodology that is designed specifically to guide the development of educational software. Therefore, in this paper, we propose a new software development methodology that provides guidance for students and contains an evaluation process that considers both the educational objectives and the role of each member in the team.

3 Literature Review

This section summarizes the strengths and weaknesses of the related works and thereby highlights the research gap and the contribution that this paper intends to make to address that gap. Generally speaking, the FYP can be considered as both an educational course and a software project [13, 14]. Hence some of the related works deal with the FYP as an educational course, whereas others view it as a software project. Therefore, in the following, the related works are classified into two groups based on the nature of the proposed model, i.e., whether it addresses the FYP as an educational course or as a software project. To collect the related works, we have used Google Scholar as the main source and used the search string "(Final Year Project, capstone, or final course) and (IT, software engineering, computer science, or computing)".

3.1 Models that address the FYP as an educational course

Benton and Radziwill [15] developed an agile learning framework based on three concepts: 1) the coevolution of the students and teachers, 2) self-management and 3) continuous improvement. In spite of the relevance of the idea, its applicability in developing educational software is not clear, i.e., the steps involved in developing a software solution are not mentioned. Rodríguez et al. [16] proposed software engi-

neering teaching model based on the scrum and which they reinforced with agile coaching. The proposed technique was compared to the rational unified process and validated using the capability maturity model integration (CMMI) framework. Validation by CMMI proved that through this model students were able to achieve higher coverage of software engineering practices. Although the Rodríguez et al. [16] model was developed to support teaching software engineering, details of its impact in terms of implementing the FYP are lacking. More recently, Fonseca and Gómez [17] presented their experience in teaching software engineering by using both problem-based learning and an agile software engineering methodology. Fonseca and Gómez [17] used real software projects as course topics. The real software projects were divided into parts and distributed to cover the course topics. Nevertheless, in their model, the evaluation of teamwork and the achievement of educational objectives are not considered.

On the other hand, Kennedy and Vossen [18] developed a scoring rubrics method for assessing teamwork in software engineering projects. They used the split-join invariance approach to split the overall team score into single student scores. Then those student scores were joined by using aggregation and averaging functions to calculate the result. However, the obvious drawback of this method is that it ignores the individual contributions of the students. In contrast, in this paper, the proposed model considers both teamwork and the individual contributions of students. Ciupe et al. [19] presented evidence on the applicability and implication of agile methodologies in education through conducting a systematic mapping study. However, they do not mention how they would overcome the difficulties of applying agile methodology in IT FYPs.

Mkpojiogu and Hussain [20] proved the applicability and usefulness of using analytical rubrics in assessing student performance in the field of software requirements engineering education. They used a four-point Likert-type rating scale to evaluate the attributes of each artefact. Although their method was successful in evaluating the whole job through evaluating each task, the evaluation of the individual effort made by each team member is not considered. Furthermore, the authors do not focus on how to manage the FYP. Yang and Yu [21] proved that dividing a software engineering class into teams of students and providing them with task-driven cases and projects could enhance the learning process. However, the measuring of the role of each individual in the teamwork task is neglected in this work as well.

3.2 Models that address the FYP as a software project

Barrella and Watson [22], Mkpojiogu and Hussain [20], and Alcarria et al. [23] proved the benefits of using rubrics in engineering projects. However, both studies lack a complete methodology. Venkataraman et al. [24] proposed 40 metrics to measure quality in IT FYPs. These metrics categorized into nine groups: software requirements, planning, design, programming practices, testing, configuration management, quality assurance and technology change management. While these metrics could help instructors to evaluate the quality of IT FYPs, they cannot be used as a software development methodology. In addition, [24] do not provide for the evaluation of

teamwork. Marques et al. [25] proposed a formative monitoring method involving reflexive weekly monitoring for use in the software engineering FYP. This monitoring method was found to improve the student learning experience. However, the study neglects to evaluate the learning objectives.

Yilmaz, et al. [26] suggested a continuous feedback and delivery mechanism for managing the life cycle of the FYP. This mechanism divides the FYP into 15 weeks, and the task for each week is identified. While beneficial, we believe that this mechanism needs to be more flexible to cope with a reality. Moreover, it does not link the FYP to education objectives. Majanoja and Vasankari [27] presented some of their reflections on the software engineering FYP and came up with five main recommendations, which were to: 1) clarify the goals of the capstone project experience, 2) highlight the importance of student commitment to the project and the team, 3) give more focus to technical studies, 4) facilitate interaction between the capstone teams and 5) provide assistant teachers and technical support. We considered these recommendations when developing our proposed model and more details on how we dealt with these recommendations can be found in section six of this paper.

Chowdhury et al. [28] investigated collaborative personality traits in undergraduate software engineering teams. They measured the roles of individuals engaged in teamwork by using metrics that reflected the activities of members in the project management online tool, Slack.com. However, the measurement of each individual's role was limited to their communication activities, which thus limited the usefulness of the measurement. Therefore, in this paper we introduce a new method for measuring the role of each individual in the whole project by classifying tasks into either individual or group tasks. Vasankari and Majanoja [29] described a framework for organizing IT capstone projects in computer science and software engineering. This framework provides students with step-by-step guidance to complete the FYP but it lacks an evaluation mechanism.

In light of the above discussion, which highlighted some of the key strengths and weaknesses of the models and frameworks proposed in the related works, we aimed to develop a software development methodology for the FYP that considered the two-fold nature of the FYP and to overcome the identified drawbacks of the existing methodologies. Firstly, by considering the FYP as an educational course, in our proposed methodology we decided to measure the success of the FYP by its ability to achieve its learning objectives. Secondly, by considering the FYP as a software project, in our proposed methodology, we decided that it was also important to provide step-by-step guidance for students to assist them in developing a successful FYP.

4 Structure and Evaluation of the Proposed Model

This section discusses in detail the structure of our proposed model for evaluating the FYP in IT programmes of study at the undergraduate level. The model combined of two methodologies: rubrics and CMMI [30]. In the current context, a rubric can be defined as "a scoring guide used to evaluate the quality of students' constructed responses" [31]. In this paper, we used CMMI as the scoring guide for evaluating the

FYP. In the following section, first the structure of our proposed model is described and then the evaluation mechanism incorporated into our proposed model is elaborated.

4.1 Structure of the proposed model

The proposed model breaks down the FYP into groups of phases, and each phase contains a number of tasks. These tasks are classified into individual tasks or group tasks. Individual tasks are executed by only one student, whereas group tasks are executed by team work (definition 1), hence the FYP can be described as a group of tasks. A phase in the FYP could be considered as a virtual container for a group of tasks and illustrates a milestone that proving completion of its tasks. A phase is completed if and only if all of its tasks are completed (definition 2). The next phase cannot be started until the previous phase is finished, i.e., the relationship between the phases is "finish to start" (definition 3). The final result of the FYP is a summation of the results of its respective phases (definition 4), while the final results of a phase is a summation of the results of its respective tasks (definition 5). Like any regular course, the FYP contains a selected set of learning objectives. These FYP learning objectives are distributed within a number of FYP tasks according to predefined percentages, and the student must achieve certain percentage scores to succeed in the project (definition 6). Hence, a project task must satisfy the set of FYP objectives in order to be considered a complete task. This means that the end of a task is achieved by satisfying its objectives by attaining a predefined acceptable percentage (definition 7).

Table 2 shows a detailed example of a FYP that explains the distribution of objectives among the tasks and phases. In this example, the FYP consists of two phases. Phase 1 consists of three tasks: task11, task12 and task13, where task11 and task13 are group tasks and task12 is an individual task. Phase 2 consists of two tasks: task21, which is a group task, and task22 which is an individual task. This FYP has five objectives that must be achieved to successfully complete the FYP. As an instance from Table 2, Task11 is designed to achieve 50% of the first objective and 20% of the third objective with percentages 20, 50 and 30, respectively. Note that each objective must be 100% satisfactory regardless of its distribution among the tasks.

The formal definitions of the structure for the proposed model are presented below:

Definition 1: $\forall$ FYP, Ph, T: FYP{Ph1{T1,…Tn},….,Phn{T1,…Tn}}, $T\begin{cases} i \\ g \end{cases}$.

Definition 2: $\forall$ Ph, T: complete (Ti) $\wedge$ Ti $\in$ Phi $\Longrightarrow$ complete (Phi)

Definition 3: $\forall$ Ph: complete (Phi) $\Longrightarrow$ start (Phi+1).

Definition 4: $\forall$ FYP, Ph: $\sum$ Ph $\wedge$ Ph $\in$ FYP $\Longrightarrow$ result (FYP).

Definition 5: $\forall$ Ph, T: $\sum$ T $\wedge$ T $\in$ Ph $\Longrightarrow$ result (Ph).

Definition 6: $\forall$ FYP, OBJ, T: $\Longrightarrow$ OBJFYP{obj1,…,objn} $\wedge$ T(obj[P]) $\wedge$ obj $\in$ OBJFYP $\wedge$ T $\in$ FYP.

Definition 7: $\forall$ T, OBJ: T {objx(P),…,objn (P)} $\wedge$ satisfy(T, obj) $\Longrightarrow$ complete(T), where FYP denotes final year project, Ph denotes phase, T denotes task, T(i) denotes individual task, T(g) denotes group task and P denotes percentage. The terms

complete () and start () are predicates that return true or false. The terms result () and satisfy () are functions that return specific values.

Table 2. Example of Allocation of Task Objectives in a Final Year Project

Phase of FYP	Task	Objective	Percentage
Phase 1	T_{11}(g)	Obj 1; Obj 3	50%; 20%
	T_{12}(i)	Obj 2; Obj 1	30%; 20%
	T_{13}(g)	Obj 4; Obj 3	100%; 50%
Phase 2	T_{21}(g)	Obj 1; Obj 5	30%; 70%
	T_{22}(i)	Obj 2; Obj 3; Obj 5	70%; 30%; 30%

The FYP in the Table 2 example has five tasks: three tasks in phase 1 and two tasks in phase 2. Thus, students should achieve five learning objectives by doing the FYP. Those five learning objectives are distributed among the five tasks. For instance, learning objective 1 is distributed among two tasks in phase 1 and one task in phase 2. In summary, the structure of our proposed model consists of three steps: define the FYP phases, divide each phase into groups of tasks, and distribute the learning objectives within the tasks by specific percentages. Table 3 describes the steps of our proposed model and output of each step.

Table 3. Describes the steps of our proposed model and output of each step

#	Step	Output
1	Define the FYP phases	Number of phases with cost and schedule definition for each phase
2	Divide each phase into groups of tasks	Description of a FYP in terms of tasks that facilities the management of a FYP.
3	Distribute the learning objectives within the tasks by specific percentages	Description of how the learning objectives could be achieved in a FYP

4.2 The evaluation process in the proposed model

Evaluation is the most significant part of any academic activity because it reflects what students have gained from the activity. Hence, the learning objectives were considered as the main evaluation target in our proposed model. As stated above, a FYP should satisfy a specific set of learning objects that are distributed among the FYP tasks. We used a rubrics technique as the evaluation mechanism so that each task could be evaluated individually. As mentioned earlier, a rubric can be defined as a set of criteria for grading assignments. The criteria that we chose for this model were adopted from CMMI, which is composed of five levels. In the following, we justify why the rubric and CMMI approaches were used in our proposed model.

Firstly, Dawson [32], Panadero et al. [33] and Fraile [34] demonstrated that rubrics are useful techniques that assist students to improve their academic output by providing them with redirection and further possibilities that are arrived via a process of self-assessment. These possibilities back to students' knowledge about the assessment which allows self-corrections to be made before a final assessment. Secondly, CMMI

is a well-established model that is used to evaluate software development companies based on their performance in previous projects. Chen et al. [35], Siju and Patel [36], Chari and Agrawal [37], and Cerdeiral and Santos [38] discussed and presented the advantages of using CMMI in software projects. However, no work (to the best of our knowledge) has used CMMI to measure academic projects. Although CMMI has been extended from measuring the quality of software companies to measuring the quality of software projects, both of these versions of the CMMI have the same structure without any modifications. However, as indicated in section two, the differences between business and educational software projects require that CMMI should be modified to cope with educational software projects. The next subsection provides an overview of the structure of CMMI. This is followed by a brief description of the changes that were made to create a modified CMMI, which we named educational CMMI.

Overview of capability maturity model integration: The five levels of CMMI are as follows:

- Level 1: Initial: The development of software in this phase could be described as chaotic. In other words, there is no standard process to follow during the development process.
- Level 2: Managed: In this level, the requirements are managed by a standard technique. However, the standard might be different from process to process.
- Level 3: Defined: This level provides more technical details about processes than the second level.
- Level 4: Quantitatively Managed: In this level, quantitative objectives, statistical methods for ensuring quality and process performance are established and used as criteria in managing processes.
- Level 5: Optimizing. This final level involves continually improving process performance through both incremental and innovative technological improvements.

Educational CMMI: In this paper, we propose using CMMI to evaluate FYP tasks. Therefore, it was crucial to adapt CMMI so that it would be suitable for application to the FYP. The modification process took place after a brainstorming workshop that included software engineering instructors and FYP supervisors. Due to the limitation of space, a description of the details of the modification process is beyond the scope of this paper. However, in short, two fundamental changes were made. First, the CMMI was reduced to four levels from five in order to create educational CMMI. In traditional CMMI, level five is an advanced level that is usually achieved by well-established IT companies. Hence it was judged that this level could be omitted without affecting the academic evaluation process. Second, the focus of each level was changed to correspond more closely with the educational project context. Primarily, CMMI was modified so that each task could be evaluated individually. The four levels of the proposed educational CMMI are outlined below:

- Level 1: Initial: At this level, students complete their task without using a predefined plan. For instance, suppose the task is writing specific code, here the students should write their code without following any standard. There are many standards

or best practices for writing code, such as standards for writing variables or standards for writing methods.

- Level 2: Managed: At this level, students should prepare preliminary requirements before starting a task. For instance, before starting a test task, preliminary test cases should have been prepared to manage the test task.
- Level 3: Defined: At this level, students should provide technical documentation that describes the task after they have completed it. For instance, suppose the task is to design an entity relationship diagram, here the students should provide a detailed technical description of each entity.
- Level 4: Quantitatively Managed: At this level, students should validate their completed task. Validation can be accomplished through applied test cases, or from feedback received from other students.

Table 4 illustrates how educational CMMI can be used for evaluating FYP by applying it to a unified example in which the task is to design a use case diagram.

Table 4. Application of Educational CMMI for Evaluating the FYP

Level	Definition	Example
Level 1: Initial	Lack of defined plan	Design a use case diagram before defining the main actors, or defining the scope of a system
Level 2: Managed	Existence of preliminary requirements	Define the roles and scope of each actor before designing the use case diagram
Level 3: Defined	Existence of technical documentation	Provide a scenario that fully describes the use case diagram
Level 4: Quantitatively Managed	Providing testing after completing a task	Test the use case diagram by running an example

In the next section, a description of the execution of our proposed model is presented.

5 Implementation of the Proposed Model

The proposed model was implemented as an experiment in the Faculty of Computers and Information Technology, where the thirty FYP were chosen as test cases for the model. Each project was assigned in two sections, first section is managed by one of tradition software development methodologies and second section is managed by our proposed model. On other words, there are thirty projects; each project is handled by two groups of students. First group is used our proposed model and another group is used one of the traditional software development methodologies. Hence, the total number of FYP involved in the experiments is sixty FYPs. Each one of the test-cases FYPs consisted of three undergraduate students. The total number of students that are involved in this experiment are 180 students. The purpose of conducting this experiment is to evaluate the performance of the proposed model by running a comparison between FYPs that are managed by our proposed model against the FYPs that are

managed by traditional SDLC models. The results of this comparison are discussed in section six.

In the following, evaluation of students' work has been presented. One of our test case FYPs has been selected as case study to explain the evaluation process. Table 5 presents a mathematical description of two phases and their respective tasks of a selected FYP.

Table 5. General Description of the Selected Phases and Their Tasks

Phase 1 {T_{11}(g), T_{12}(i), T_{13}(g)} Phase 2 {T_{21}(g), T_{22}(i)} Where: T_{nm}(x) means T represents task, n represents phase number, m represents task number, and where X$\begin{cases} i \\ g \end{cases}$, i is for the individual and g is for the group

As shown in Table 5, phase 1 of a FYP consisted of three tasks, where the first and third tasks are group tasks and the second task is an individual task. Phase 2 contained two tasks, where the first task is a group task and the second task is an individual task. Each task was evaluated by educational CMMI, which means that the FYP was rated from 1 to 4; with 1 being the lowest score. In the following, the values of the above two phases and their respective tasks are presented in more detail.

Phase 1 consisted of performing an "analysis", which required the completion of task11 "design a flowchart", task12 "design a data flow diagram" and task13 "write a software requirement specification". Phase 2 "Design", consisted of task21 "design a use case diagram" and task22 "design a class diagram".

The three learning objectives that should be attained by completing these phases and their respective tasks were as follows:

- Learning objective 1: Demonstrate usage of analysis methods
- Learning objective 2: Demonstrate usage of design methods
- Learning objective 3: Demonstrate teamwork skills

The percentage distribution of each of the objectives among the five tasks was as follows:

- Objective 1 is completely satisfied (100%) by achieving 25% in $task_{11}$ and in $task_{12}$, and 50% of $task_{13}$.
- Objective 2 is completely satisfied (100%) by achieving 50% in $task_{21}$ and in $task_{22}$.
- Objective 3 is completely satisfied (100%) by achieving 30% in $task_{11}$, 30% $task_{13}$, and 40% $task_{21}$.

Equations (1), (2) and (3) describe the satisfaction of objectives 1, 2 and 3, respectively:

$$Obj_1\,(100\%) = \{Task_{11}(25\%), Task_{12}(25\%), Task_{13}(50\%)\} \quad (1)$$

$$Obj_2\,(100\%) = \{Task_{21}(50\%), Task_{22}(50\%)\} \quad (2)$$

$$Obj_3\ (100\%)\ =\ \{Task\ _{11}(30\%), Task\ _{13}(30\%), Task\ _{21}(40\%)\} \quad (3)$$

In our proposed model, teamwork was measured through the evaluation of the group tasks. In other words, teamwork was evaluated by summation of all the group tasks. The reason that we decided to define the teamwork evaluation as the summation of all the group tasks is because the nature of the group tasks should reflect the efforts made by all the team members. Therefore, a group task was evaluated by using the average of the summation of the evaluations of all the students in the group. Definition 8 shows the equation used for evaluating group tasks:

Definition 8: ∀ Stu, Tg: Avg ($\sum result(Stu, Tg)$) $\Longrightarrow$ result(Tg)

Where Stu denotes student, Tg denotes group task, eva() is a function that returns the evaluation of a student in a group task, Avg() is a function that returns the average of the summation of the evaluations of all the students in the group, and result() is a function that returns the evolution of the group task.

Table 6 shows the evaluation values for the tasks of one of our test-case FYPs. These values were obtained according to the description of educational CMMI.

Table 6. The Evaluation Values for the Tasks of one of the Test-case FYPs

Phase	Task	Stu1	Stu2	Stu3
Phase 1 (Analysis)	T11(g) Design a flowchart	3		
		4	2	3
	T12(i) Design a data flow diagram	3	2	2
	T13(g) Write a software requirement specification	2.3		
		2	3	2
Phase 2 (Design)	T21(g) Design a use case diagram	2.7		
		3	3	2
	T22(i) Design a class diagram	3	2	2

From Table 6, it can be seen that students were able to achieve four, three, or two marks for each task. A mark for each task was awarded based on the following criteria:

- Four marks: Where a student satisfied preliminary requirements, provided a technical description, and provided validation, they were awarded four marks for the task. Validation could be implemented by creating a scenario, for example, or running case.
- Three marks: Where a student satisfied preliminary requirements and provided a technical description, they were given three marks.
- Two marks: Where a student was only able to satisfy preliminary requirements, they were awarded two marks.

The final values for the group tasks were taken as the average of the individual students' marks. We adopted this calculation methodology as we considered that taking the average as the value of the group task would reinforce the collaboration between

students, which would in turn lead to strengthening the teamwork skills of each member of the group.

In the next section, the evaluation of our proposed model is presented and discussed and some concluding remarks are made.

6 Discussion and Conclusion

The applicability of our proposal model has been evaluated by experiment that involved sixty FYP. Each FYP implemented by group of three students.

As has been mentioned in section five, each FYP project was assigned to two groups. The first group managed the FYP by using our proposed model, whereas the second group managed the FYP by using the traditional SDLC. The two groups were asked to complete an online questionnaire in order to gain their feedback on the two methodologies. The questionnaire contained the following statements:

1. Working on this FYP improved my teamwork skills.
2. Working on this FYP improved my technical skills.
3. Working on this FYP taught me how to manage projects tasks.
4. Working on this FYP taught me how to control a project.
5. Working on this FYP helped me to understand learning outcomes.

The students were asked to respond to the statements on a Likert-type rating scale ranging (1: strongly disagree, 2: disagree, 3: don't know, 4: agree, 5: strongly agree). Figure 1 shows the feedback received from the two groups of students, where group 1 was the group that completed the FYP by using our proposed model and group 2 completed the FYP by using the traditional SDLC. It is obvious from the figure that group 1 was more satisfied with the model they had used as compared to group 2, particularly in regards to statements 1, 3, 4, and 5.

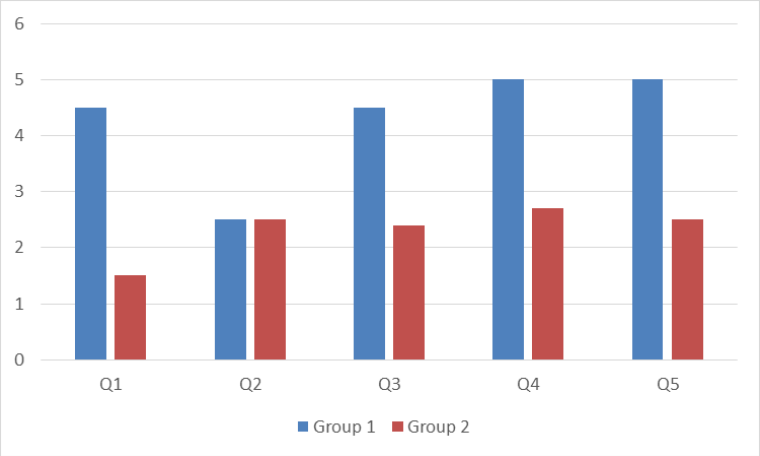

Fig. 1. Feedback from the two groups of students.

The students' feedback therefore indicated that our proposed model was successful from the students' perspective. With regards to the feedback received for statement 2, which was equal between the two groups, this indicated that our proposed model did not support the students in improving their technical skills. This may be due to the absence of a special task that focused on technical skills. This could be addressed by including a task such as prepare a software tool or test the implementation environment. Hence, this feedback provides room for further research.

Also, the students in group 1 were asked an additional question: What difficulty did you have in using development model? The answers they gave were: 1) finding suitable time for all members, 2) variation in technical skills, and 3) personnel conflicts. The answers given to this question drew our attention to the importance of defining the factors for team selection, and this represents one direction we will follow in our future work.

The other key finding of the comparison between the two groups was that the first group achieved higher marks than the second group in the final mark awarded for the FYP. Figure 2 shows the comparison of results average between the two groups. Evaluation of our proposed model has been achieved by direct and indirect assessment. Direct assessment is obtained from final results of the experiment FYPs, and indirect assessment is obtained by questionnaire's results.

According to this direct and indirect assessment, it can be stated that our proposed model could be more successful in managing educational FYP than traditional SDLC.

On the other hand, it is plausible that our proposed model could provide a deeper analysis of student performance than the SDLC methodology by, for example, using it to examine the relationship between a student's effort as an individual and as a group member. This will therefore be a second direction that we will follow in our future work.

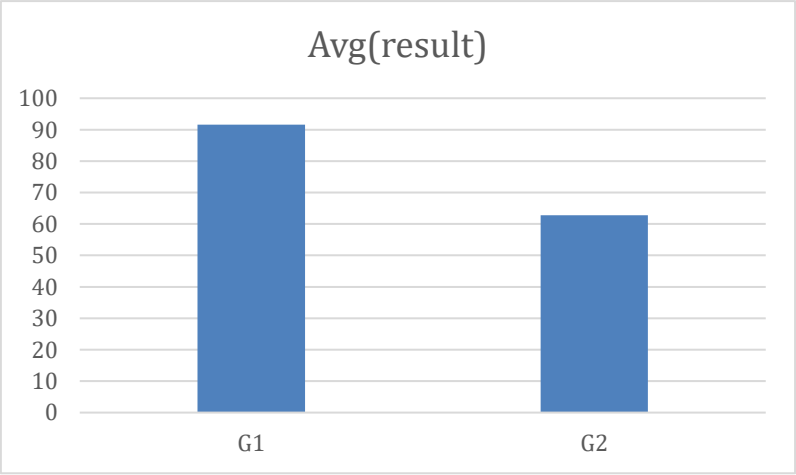

Fig. 2. The comparison of results average between the two groups.

In conclusion, the contribution of this paper is threefold. The first contribution being highlighting the differences between business and educational software, and that these differences necessitated the development of a new model tailored specifically to educational software. The second contribution was our modification of CMMI to produce educational CMMI, which was then used as part of a rubric for evaluating students' academic output. The third contribution was the proposed new model for managing the FYP in undergraduate IT and IT-related programmes of study. This proposed model was evaluated by academic results and by student feedback and the results showed that the proposed model was able to achieve its targeted outcomes. From previous discussion the contribution could be summarized as:

- Proving the inappropriateness of the software development methodologies which are used currently in developing FYP.
- Modifying CMMI to be used for educational measurements. One of the biggest pedagogical challenges in FYP is measuring the effort, or contribution of individual in a teamwork task. Our proposed educational CMMI could handle this challenge.
- Proposed new model for developing educational software.

As future work, we are planning to apply the proposed educational software development model in IT postgraduate studies.

7 Acknowledgement

The authors are thankful to the Deanship of Scientific Research (DSR), the University of Tabuk, Saudi Arabia, for financial support under the grant number S-0084-1439

8 References

[1] Sommerville, L. (2015). Software Engineering (10th Edition). Pearson.
[2] Elliott, G. (2004) Global Business Information Technology: an integrated systems approach. Pearson Education. p.87.
[3] Robert, C. (2019). Clean Architecture: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series). Prentice Hall.
[4] Whitten.J., Bentley,L. (2005). Systems Analysis and Design Methods. 7th edition. ISBN-10: 0073052337. McGraw-Hill/Irwin.
[5] Tan, J., Phillips, J. (2005). Incorporating service learning into computer science courses. Journal of Computing Sciences in Colleges (JCSC) vol 20 (4).
[6] Longmuss, J., Höhne, B., Oberländer, A. (2016). Agile learning: Bridging the gap between industry and university A model approach to embedded learning and competence development for the future workforce. 44th SEFI Conference, Tampere, Finland.
[7] Jacques, S., Bissey, S., & Martin, A. (2016). Multidisciplinary Project Based Learning Within a Collaborative Framework: A Case Study on Urban Drone Conception. iJET, 11, 36-44. https://doi.org/10.3991/ijet.v11i12.5996

[8] Tatyana I. Anisimova, Fairuza M. Sabirova, Olga V. Shatunova. (2020). Formation of Design and Research Competencies in Future Teachers in the Framework of STEAM Education. International Journal of Emerging Technologies in Learning (iJET). Vol 15, No 02. https://doi.org/10.3991/ijet.v15i02.11537

[9] Sosa, E., Salinas, J., & De Benito, B. (2019). Emerging technologies (ETs) in education: A systematic review of the literature published between 2006 and 2016. International Journal of Emerging Technologies in Learning, 12 (5), p. 128-149. https://doi.org/10.3991/ijet.v12i05.6939

[10] Zhanat Nurbekova, Vadim Grinshkun, Gaukhar Aimicheva, Bakyt Nurbekov, Kalima Tuenbaeva. (2020). Project-Based Learning Approach for Teaching Mobile Application Development Using Visualization Technology. International Journal of Emerging Technologies in Learning Vol 15, No 08. https://doi.org/10.3991/ijet.v15i08.12335

[11] Project Management Institute PMI (2017). A Guide to the Project Management Body of Knowledge (PMBOK® Guide)–Sixth Edition. https://doi.org/10.1109/ieeestd.2011.6086685

[12] Sadowski, C., Zimmermann, T. (2019). Rethinking Productivity in Software Engineering. Apress.

[13] Garousi, V. (2010). Applying peer reviews in software engineering education: an experiment and lessons learned. IEEE Transactions on Education, VOL. 53, NO. 2. https://doi.org/10.1109/te.2008.2010994

[14] Ma, K., Teng, H., Du, L., & Zhang, K. (2014). Project-driven learning-by-doing method for teaching software engineering using virtualization technology. International Journal of Emerging Technologies in Learning, 9. https://doi.org/10.3991/ijet.v9i9.4006

[15] Benton, M. C., & Radziwill, N. M. (2011). A path for exploring the agile organizing framework in technology education. Proceedings - 2011 Agile Conference, Agile 2011, 131–134. https://doi.org/10.1109/agile.2011.51

[16] Rodríguez, G., Soria, A., Campo, M. (2015). Supporting Assessment of Practices in Software Engineering Courses, IEEE Latin America Transactions, VOL. 13, NO. 9. https://doi.org/10.1109/tla.2015.7350070

[17] Fonseca, V.M.F., Gómez, J. (2017) Applying Active Methodologies for Teaching Software Engineering in Computer Engineering. The IEEE Journal of Latin-American Learning Technologies (IEEE-RITA) vol. 12, no. 3, pp. 147-155. https://doi.org/10.1109/rita.2017.2738178

[18] Kennedy, I., Vossen, P. (2017). Software engineering teamwork assessment rubrics: combining process and product scoring. DeLFI 2017, Germany.

[19] Ciupe, A., Ionescu,R., Meza, S., Orza, B. (2018). Towards Agile Integration within Higher Education:A Systematic Assessment, BRAIN. Broad Research in Artificial Intelligence and Neuroscience, VOL. 9 NO 3.

[20] Mkpojiogu, E., Hussain, A. (2017). Assessing Students' Performance in Software Requirements Engineering Education Using Scoring Rubrics The 2nd International Conference on Applied Science and Technology 2017 (ICAST'17). Kedah, Malaysia. https://doi.org/10.1063/1.5005425

[21] Yang, Y., Yu, D. (2019). Task-driven Teamwork Teaching Strategies in Software Engineering, International Conference on Education Reform, Management Innovation and Social Science (ERMISS 2019), Beijing, China.

[22] Barrella, E. & Watson, M.K. (2016) "Developing a cross-disciplinary sustainable design rubric for engineering projects", 8th Conference on Engineering Education for Sustainable Development, Bruges, Belgium.

[23] Alcarria, R., Bordel, B., & de AndrÃ, D. M. (2018). Enhanced peer assessment in MOOC evaluation through assignment and review analysis. International Journal of Emerging Technologies in Learning (iJET), 13(1), 206-219. https://doi.org/10.3991/ijet.v13i01.7461

[24] Venkataraman, S., Al Hussein, A., Siddappa, M. (2017) Development of software metrics for improving the quality of the under graduate student projects in computer science /information science/information technology/computer engineering. International Journal of Computer Science and Network Security, VOL.17 No.10,

[25] Marques, M., Ochoa, S.F., Bastarrica, M.C., Gutierrez, F.G (2018). Enhancing the Student Learning Experience in Software Engineering Project Courses. IEEE Transactions On Education, Vol. 61, No. 1. https://doi.org/10.1109/te.2017.2742989

[26] Yilmaz M., Tasel F.S., Gulec U., Sopaoglu, U. (2018). Towards a process management life-cycle model for graduation projects in computer engineering. PLoSONE 13(11):e0208012. https://doi.org/10.1371/journal.pone.0208012

[27] Majanoja, A-M. and Vasankari, T. (2018). Reflections on Teaching Software Engineering Capstone Course.InProceedings of the 10th International Conference on Computer Supported Education (CSEDU 2018) - Volume 2, pages 68-77. https://doi.org/10.5220/0006665600680077

[28] Chowdhury, S., Walter, C., Gamble, R. (2018). Toward increasing collaboration awareness in software engineering teams. FIE 2018: San Jose, CA, USA. https://doi.org/10.1109/fie.2018.8659198

[29] Vasankari T., Majanoja AM. (2019) Practical Software Engineering Capstone Course – Framework for Large, Open-Ended Projects to Graduate Student Teams. In: McLaren B., Reilly R., Zvacek S., Uhomoibhi J. (eds) Computer Supported Education. CSEDU 2018. Communications in Computer and Information Science, Vol 1022. Springer, Cham. https://doi.org/10.1007/978-3-030-21151-6_16

[30] CMMI Product Team. (2010), Improving processes for developing better products and services. CMMI for Development, Version 1.3. Technical report. Software Engineering Institute. Carnegie Mellon University.

[31] Panadero, E., Jönsson, A.(2013). "The use of scoring rubrics for formative assessment purposes revisited: A review". Educational Research Review. 9: 129–144. https://doi.org/10.1016/j.edurev.2013.01.002

[32] Dawson, Ph. (2015). "Assessment rubrics: towards clearer and more replicable design, research and practice Phillip". Assessment & Evaluation in Higher Education. 42 (3): 347–360. https://doi.org/10.1080/02602938.2015.1111294

[33] Panadero, E., Jonsson, A., & Strijbos, J. W. (2016). Scaffolding self-regulated learning through self-assessment and peer assessment: Guidelines for classroom implementation. In D. Laveault & L. Allal (Eds.), Assessment for Learning: Meeting the challenge of implementation (pp. 311-326). New York: Springer. https://doi.org/10.1007/978-3-319-39211-0_18

[34] Fraile, J., Panadero, E., Pardo, R. (2017). "Co-creating rubrics: The effects on self-regulated learning, self-efficacy and performance of establishing assessment criteria with students. Studies in Educational Evaluation. 53: 69–76. https://doi.org/10.1016/j.stueduc.2017.03.003

[35] Chen, J. J., Su, W. C., Wang, P. W., & Yen, H. C. (2013). A CMMI-based approach for medical software project life cycle study. SpringerPlus, 2(1), 266. https://doi.org/10.1186/2193-1801-2-266

[36] Siju, H. L., & Patel, P. (2017). Survey of Models and Tools for Project Monitoring and Control. International Journal of Science Technology & Engineering, (3), 10.

[37] Chari, K., & Agrawal, M. (2018). Impact of incorrect and new requirements on waterfall software project outcomes. Empirical Software Engineering, 23(1), 165-185. https://doi.org/10.1007/s10664-017-9506-4

[38] Cerdeiral, C. T., & Santos, G. (2019). Software project management in high maturity: A systematic literature mapping. Journal of Systems and Software, 148, 56-87. https://doi.org/10.1016/j.jss.2018.10.002

9 Authors

Abdelrahman Osman Elfaki is member of Information Technology Department, Faculty of Computers and Information Technology at University of Tabuk.

Zaid Bassfar is associate professor at Information Technology Department, Faculty of Computers and Information Technology at University of Tabuk. Currently, Dr. Zaid is a dean of Deanship of Admission and Registration at University of Tabuk. zbassfar@ut.edu.sa

Article submitted 2020-05-03. Resubmitted 2020-06-25. Final acceptance 2020-07-07. Final version published as submitted by the authors