TOWARDS INCORPORATING HUMAN FACTORS IN THE SOFTWARE PROJECT COST CONTROL MODELS

TARIG A KHALID, PHD.
ENG-THIAM YEOH, PHD.

MULTIMEDIA UNIVERSITY - MALAYSIA

Abstract: Controlling software cost is crucial to complete projects within the planned budgets. Unfortunately, the existing software cost control models; especially the earned value management system (EVM) experience some limitations. These models, mainly focus on factors such as time and cost while ignoring the related human factors. Since software is delivered by people and affected by their competencies, it is expected that their associated attributes would be quantified and incorporated in such models. Although many papers discussed the human factor issues, however, very few suggested quantified indices to model such factors. This paper aims at introducing the Developer Competency Index (DCI) and Team Competency Index (TCI) as a quantification of the human skills needed by the software engineers. Function Point Analysis (FPA) is extensively used in the calculation process. Moreover, using data drawn from five actual projects, the study shows how these indices are related to the cost of software reworks and how they can be used to monitor and control the team performance throughout the project lifecycle.

KEYWORDS: HUMAN FACTORS, SOFTWARE COMPETENCY, SOFTWARE COST CONTROL, EARNED VALUE MANAGEMENT, EVM

DOI NUMBER: 10.19255/JMPM01704

1. Foreword

The increasing demands in the various types of software applications have led to the initiation of a huge number of software development projects. In concurrence with these demands, software costs are increasingly large (Boehm, 1981). Unfortunately, a considerable percentage of the software projects experience cost overruns. The Standish Group reported that the success rate of software development has just increased from 16% to 32% during the period from 1994 to 2013 (Chaos Manifesto, 2013). Yet, the cost overrun is still on the increase. A joint-research conducted by McKinsey and Oxford reported that the large software projects, with budgets exceeding 15 million USD, were severely challenged due to 66% of cost overrun (Bloch, 2012; Chandrasekaran, 2014). These amounts of cost-overrun indicate that developing software systems is both expensive and difficult. This is due to a series of problems such as poor or volatile requirements, poor project management, low skilled developers, and cost and schedule overruns (Boehm, 1981; Bennatan, 2000). Reworks make managing software projects severely challenging. In fact, 40% to 50% of the development efforts spent in reworks and maintenance that may approach 70% of the total lifecycle cost (Boehm and Basili, 2001; Efe and Demiros, 2013). In comparison with the other IT projects, software projects incur the highest risk (Bloch, 2012).

To address the problem of cost overrun, software project managers use various project management tools and techniques. The most recognized cost control technique provided by the project management best practices is the Earned Value Management (EVM). It is "the only one of many measures" which is extensively used to assess the status of a software development project (NAVAIR, 2004). The EVM controls project performance by tracking the schedule and cost via two indicators; the Schedule Performance Index (SPI) and Cost Performance Index (CPI) (PMI, 2017).

These indicators are calculated at many checkpoints during the project lifecycle. If CPI is found below 1 this indicates a problem of cost overrun while the project is considered cost-healthy if CPI is equal to or greater than 1. Also, SPI<1 indicates project delays, SPI=1 indicates that the project is expected to finish as planned, and SPI>1 means that the project will finish earlier than planned. Moreover, the EVM has a forecasting scheme for the overall cost at the end of the project. It calculates the estimate at complete (EAC) in terms of the CPI and SPI (PMI, 2017) (see Appendix A for more details). Unfortunately, the EVM model has some limitations in controlling software projects, especially the inaccuracy in forecasting the overall cost at the project completion. The SPI and CPI may positively evaluate the schedule and cost efficiencies while they have nothing to report about the quality of the software product itself. EMV measurements are based on neither product metrics nor product quality (Solomon, 2005). EVM has no quality indicator as it only measures time and cost and accepts the deliverables upon their completion (Ghosh, 2015). It measures the quantity, not the quality (Solomon, 2005).

One of the major issues related to the EVM is that it cannot estimate the cost of software reworks that occur after testing (Efe and Demiros, 2013). However, unless the existing project management tools and techniques have been modified, the special needs of the software projects will continue unsatisfied (Efe and Demiros, 2013).

On the other hand, software is developed by people, not machines. Software development is a human-centric activity and highly influenced by the psychology, emotions, and personality variables of the developers, which play a determining factor in their work styles (Shneiderman, 1980; AlQaisi et al., 2013; AlQaisi et al., 2016; AlQaisi et al., 2017). The quality of the software product is affected by developers' personal attributes such as their level of knowledge, skills, cultural differences, and diverse attitudes (Mishra and Mishra, 2011) (Colomo-Palacios et al., 2013; Colomo-Palacios et al., 2014). In fact, the major issues in software development are human-related rather than technical ones and the major software system failures are attributed to human issues (Waychal and Capretz, 2017; DeMarco and Lister, 2013). According to a survey conducted by Yilmaz et el. (2017), a significant number of software projects failed due to social issues. Despite the huge impact of the human factors on the software development process, they are always overlooked and not well-studied in comparison with the technical factors (Waychal and Capretz, 2017; DeMarco and Lister, 2013).

Unfortunately, the existing cost control tools such as the EVM do not incorporate such human factors into their models.

The objective of this research is to answer the following research questions:

RQ1: Can we establish a unified software development competency framework?

RQ2: Can we construct quantified competency indices for the individual developers and the whole team?

RQ3: Can we incorporate these indices into the software cost control tools such as the EVM?

RQ4: Can we monitor and control these indices as we used to do with the schedule and cost?

By answering these questions, the study introduces human related indices to a software project cost control model; namely the Developer Competency Index (DCI) and Team Competency Index (TCI). These indices would be incorporated into the EVM model to enhance its ability in both assessing the project performance and forecasting software cost at project completion. The applicability of the TCI is tested by making use of a sample dataset collected from 5 actual projects. The possibility of using the TCI along with the other cost control variables is discussed.

The rest of the paper is organized as follows; Section 2 covers the literature review. Section 3 describes the research methodology while section 4 provides a detailed description of the proposed framework and the quantified model. The fifth section shows the evaluation of the proposed model. Section 6 covers the research discussions. Conclusions and future work are discussed in Section 7.

2. Literature review

Software project cost is affected by many technical and non-technical factors. However, for decades, researchers have focused on the study of the technical factors. Technical factors such as the lack of quality performance indicators, the emergence of the agile development methodology, uncertainty in performance measurements, requirements volatility, and uncertainty associated with the project risks have been extensively investigated. These researches resulted in various extensions to the EVM such as the quality EVM, agile EVM, fuzzy EVM, and performance-based EVM (Ju and Xu, 2017; Khalid and Yeoh, 2015; de Souza et al., 2014; Naeni et al., 2014; Xu et al., 2010; Solomon, 2005). Although these researches have created a rich body of knowledge, they did not well-cover the non-technical factors that may affect the effectiveness of the software cost control tools.

2.1 Human issues in the software development process

Recently, researchers started to focus on the impact of the human factors, such as the impact of the developers' work habits and personality traits on the productivity patterns (Meyer et al., 2017a). In this context, Meyer et al. (2017b) identified six groups of developers, according to their perceptions of productivity. These groups are based on the personal attributes of the developers, namely; social, lone, focused, balanced, leading, and goal-oriented developers. They argued that taking this grouping into account during planning and team building will increase the developers' productivity. Moreover, measuring the individual attributes of the developers and grouping them according to their types of personality will positively impact the software development teams (Anderson et al., 2018). Building the teams based on the human aspects as well as monitoring the team climate can promote the developers' satisfaction and improve the product quality (Acuña et al., 2015). In general, the contemporary software development methodologies, especially the agile ones paid more attention to the human factors (Broza, 2012; AlQaisi, 2017; Schwaber and Sutherland, 2017). However, while many researchers studied the importance of the human factors and their impacts on the developers' productivity and the product quality, unfortunately, a very limited number of

studies seem to provide human-related frameworks or quantified indices.

Among these few studies, we found that Sedelmaier and Landes (2014) established a framework named as the Software Engineering Body of Skills (SWEBOS) defining the soft skills needed by the software engineers. The SWEBOS suggested six main areas, namely; collaboration, communication, structure, personal competencies, consciousness of problems, and the competence to solve problems. The collaboration area covers the skills needed to collaborate with other developers to handle problems. The communication area describes the skills needed to collaborate with other developers to handle problems while the structure covers the skills needed to collaborate with other developers to handle problems in relation to the assigned tasks. The personal competencies show how to handle challenges in a goal-oriented manner. The consciousness of problems covers the capability of the software developer to comprehend complex processes and systems, and their mutual relationships. The competence to solve problems describes how software developers are capable to apply their knowledge and skills to solve problems. However, the SWEBOS did not suggest a model to quantify the listed skills. Also, it did not include the related hard skills as it only focused on the soft skills. However, the hard skills are well-covered by the third version of the Software Engineering Body of Knowledge (SWEBOK), which suggested 15 knowledge areas relevant to the discipline of software engineering (Bourque and Fairley (eds.), 2014). Among these knowledge areas, five areas are considered as core. These are software requirements, software design, software construction, software testing, and software maintenance.

On the other hand, Colomo-Palacios et al. (2013) suggested a framework to evaluate the competencies of the software developers. It consists of generic and technical competencies. They extracted the generic competencies from the Spanish white book for university degrees in computer science (Casanovas et al., 2004).

These generic competencies included, but not limited to, creativity, leadership, interpersonal skills, and critical thinking. On the other hand, the technical skills are extracted from the SWEBOK including the five core processes in addition to the software quality, configuration management, software engineering management, software engineering process, and software tools and methods. Moreover, they provided a scale to evaluate each competence. This scale is described as low, medium, high and very high. Accordingly, they suggested that the evaluation of each developer should be done by the supervisors, peers, and subordinates. The result will be the average of these evaluations. The contribution of this research is the establishing of a unified competency framework quantifying the individual competences of the developers. However, it didn't integrate the individual competences into a unified index describing the whole competences of the individual developers. Moreover, the suggested framework is well-structured into categories containing the related competencies. Instead, it provides a long list of individual competences, which makes it difficult to monitor and assess these competencies at a higher level.

2.2 Human issues in the EVM model

With respect to the EVM enhancements, Vargas (2004) suggested the use of the EVM to model the performance of the members of the project team. This is achieved through an introduction of a new index called Human .Pracharasniyom et al. (2015) suggested an extended EVM model. The suggested model calculates the planned and actual values of the work performed by a project member. Accordingly, they suggested a human resource performance index (HRPI) indicating whether the project member can perform the assigned tasks as planned or not. Performance Index (HPI). The main shortcoming is that the HPI is expressed in terms of the main indices of the traditional EVM without incorporating real human factors into the model. The basic parameters of the model are the Human Value (HV), the Earned Value (EV) as defined by the traditional model, and the Human Earned Value (HEV). The HV is calculated in terms of the actual cost (AC) as the measurement of the labor ability in comparison with his/her actual costs. HV can be less than, greater than, or equal to 1. Then, HEV is given by multiplying HV and EV. However, the process of assigning a value to HV is not well-described in the study. Moreover, the study did not suggest human factors upon which the value of HV would be defined. In fact, the suggested model expressed the human resource (HR) related indices in terms of the traditional cost and schedule indices.

As a conclusion, our literature review indicates a huge knowledge gap in quantifying human factors as well as incorporating these factors in the software cost control tools, especially the EVM. Yet, the software engineering literature has only limited publications researching into the importance of the human-side of the software development (AlQaisi, 2017; DeMarco and Lister, 2013; Broza et al., 2012).

3. Methodology

To answer the research questions, we follow the subsequent steps:

STEP1- Building a unified competency framework: This framework consists of the technical and nontechnical skills as needed by the software developers. This is achieved through literature review and synthesis, which results in modifying and integrating the SWEBOK and SWEBOS.

STEP2 – Developing a quantified competency model: We develop this model by constructing the DCI and TCI indices as follows:

1. We assign relative weights to each element in the competency framework.
2. Then, we develop an equation to calculate the DCI for each developer, according to the assigned values of the framework elements.
3. We calculate the size of the software coded by each developer. This is done using the function point analysis (FPA) method. The size is calculated in function points (FPs). We adopt the FPA since it provides measurements independent of the development tools used. Moreover, it enables calculating the software size from the given list of requirements.
4. Accordingly, we calculate the share of each developer by dividing the size developed by him/her by the total size of the software. This value is called the impact factor (IF) of the developer.
5. Finally, we calculate the overall TCI by summing up the multiplications of the DCI and IF for all developers.

STEP 3 – Model evaluation: The model is to be evaluated as follows:

1. We collected data from real projects. These data consist of the developers' competency data, project cost performance data, and the FPA data, which present the software size developed and the share of each developer.
2. For each project, we calculated the TCI and the variables related to the cost performance information.
3. We tested the correlation between the TCI and each of the cost variables. A strong correlation of the TCI with at least one of the cost related variables may indicate the possibility of incorporating the TCI into the EVM model.
4. We illustrate the possibility of monitoring and controlling the TCI during the lifecycle of the projects.

The research steps and its relation to answering the research questions are shown in **Figure 1**. The detailed descriptions of the three steps are shown in the two subsequent sections.

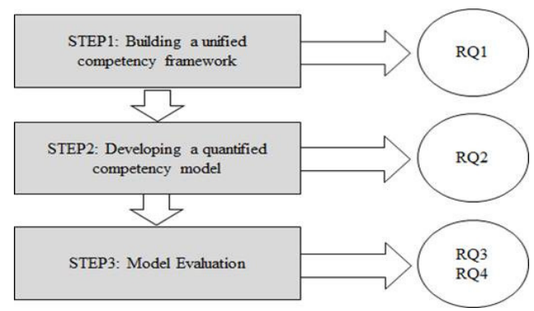

FIGURE 1: Research logical flow

4. Proposed framework and model

4.1 Software development competency framework (SDCF)

We suggest the SDCF as a framework, including the technical and non-technical skills need by the software developers. The SDCF is nothing but a customized combination of the SWEBOK and SWEBOS. The technical skills are extracted from the SWEBOK while the non-technical skills are extracted with modifications from the SWEBOS.

The technical skills include the SWEBOK core knowledge areas, except the software maintenance as it usually happens during operations i.e. after the project completion. Each knowledge area constitutes a competency. Moreover, we add two new competencies. The first one is about the big picture of the software development lifecycle, namely; the capability of selecting and adapting the appropriate software development methodology. The second one, is called the system knowledge, which covers the hard skills other than the software development, but still needed by the developers to accomplish their software development tasks. These include knowledge related to the operating systems, database management systems (DMBS), and information security. In general, each competency covers a set of criteria as illustrated in **Table 1.**

ID	Competency	ID	Criteria
C_1	Software Requirements	C_{11}	Acquiring the requirements engineering concepts
		C_{12}	Capable of eliciting requirements
		C_{13}	Capable of analysing requirements
		C_{14}	Capable of conducting requirements trade-off analysis
C_2	Software Design	C_{21}	Acquiring the software design concepts
		C_{22}	Acquiring the necessary skills related to the design tools used
		C_{23}	Capable of conducting design trade-off analysis
		C_{24}	Acquiring accreditations related to the design tools used
C_3	Software Construction	C_{31}	Acquiring the software construction concepts
		C_{32}	Acquiring the necessary skills related to the development tools used
		C_{33}	Acquiring accreditations related to the development tools used
C_4	Software Testing	C_{41}	Acquiring the software testing concepts
		C_{42}	Acquiring the necessary skills related to unit testing
		C_{43}	Acquiring the necessary skills related to integration testing
C_5	Development Lifecycle	C_{51}	Acquiring the necessary knowledge related to the software development lifecycle
		C_{52}	Capable of adapting the software development activities to the adopting methodology
C_6	System Knowledge	C_{61}	Acquiring the necessary skills related to the data base management systems (DBMS) used
		C_{62}	Acquiring the necessary knowledge related to the operating system environment
		C_{63}	Acquiring the necessary knowledge related to the information security

TABLE 1: SDCF technical competencies

On the other hand, the nontechnical skills include the SWEBOS skills with some modifications. The communications and personal competencies are combined into one category called communication skills. Moreover, we combine the "consciousness of the problem" and the "competence to solve problems" into one category named problem-solving. The details are illustrated in **Table 2.** In **Table 1** and **Table 2**, the competencies and their associated criteria are identified with unique IDs to ease referencing and traceability.

4.2 Developer competency index (DCI)

In our literature review, we didn't find an index quantifying the developer's competency. This study suggests the DCI as an index quantifying the whole elements of the SCDF. The SDCF consists of 10 competencies from C1 to C10. Moreover, we suggest a relative weight for each competency such that competency Ci has a specific weight Wi according to its importance within the project context. The scale of Wi is as follows:

Wi = {very low, low, average, high, very high} (1)

Where very low =1, low = 2, average = 3, high = 4, and very high = 5.

ID	Competency	ID	Criteria
C_7	Collaboration	C_{71}	Cooperating with others in a team
		C_{72}	Willing to communicate with others outside the team
		C_{73}	Capable of and willing to fit themselves into given structural and process organizations.
		C_{74}	Appreciates the professional competencies of others and behave with respect.
C_8	Communication Skills	C_{81}	Capable of handling criticism
		C_{82}	Capable of presenting their own ideas and issues and expressing subjects related to their domain of knowledge and expertise to others.
		C_{83}	Capable of conducting meetings, interviews, and workshop necessary to manage requests for change
		C_{84}	Capable of resolving conflicts constructively.
C_9	Self-Structure	C_{91}	Capable of analytic thinking
		C_{92}	Capable of motivating themselves, even in complex situations, and over an extended period.
		C_{93}	Capable of accepting responsibilities and of solving problems independently and in a self-directed manner.
		C_{94}	Capable of planning and executing their tasks a realistic and organized manner.
C_{10}	Problem Solving	C_{101}	Capable of abstracting and modelling complex situations.
		C_{102}	Capable of developing creative potential solutions for professional problem settings.
		C_{103}	Capable of evaluating different approaches/solutions

TABLE 2: SDCF nontechnical competencies

Moreover, each criterion within a competency, has a relative weight from 1 to 5, according to the following ratings; very low, low, average, high, and very high respectively.
Since the maximum rate for each criterion, in category Ci, is 5, then the maximum rating Maxi is equal to 5 multiplied by j, where j is the number of criteria within Ci.
Finally, the developer competency index for the individual developer (DCI) is given by:

$$DCI = \frac{\sum_{i=1}^{10}[W_i \times \sum_{j=1}^{N}(\frac{C_{ij}}{5 \times j})]}{\sum_{i=1}^{N} W_i} \quad (2)$$

The scale used in quantifying the elements of the SDCF is similar to the one suggested by Colomo-Palacios et al. (2013) with a simple modification of adding the scale "very low".
To calculate the DCI for a specific developer, each element should be assigned a value on the scale of 1 to 5. We suggest that these values are filled by the project managers, team leaders, and human resource personnel in the software firm. Their collective judgement is expected to minimize the bias incurred by the individual judgement. Alternatively, the values may be filled by the individuals in an independent manner and the average will be calculated as suggested by Colomo-Palacios et al. (2013).

4.2 Developer competency index (DCI)

We suggest a new index called TCI to represent the competency of the whole team. It resembles the centroid of the individual team members. The value of the TCI depends on two factors; the first one is the DCI of each developer while the second one is the percent software size developed by each developer. This percentage represents the impact or the contribution of the individual developer to the project. We suggest a variable called the Impact Factor (IF) to represent the developer's contribution.
To calculate IF, we suggest the use of the Function Point Analysis (FPA) in calculating the software size. We use the FPA as described by IFPUG (2010). A brief description of the FPA is found in **Appendix B**.

The impact factor of Developeri is given by:

$$IFi = \frac{FP_i}{FP_{total}} \quad (3)$$

Where FPi is the size of the software developed by Developeri and FPtotal is the total size of the software developed by the whole team. Given that the sum of the impact factors for the whole team is equal to 1, TCI is given by:

$$TCI = \sum_{i=1}^{N} DCIi \text{ x } IFi \quad (4)$$

Where DCIi is the competency index of Developer(i), IFi is his/her impact factor, and N is the number of developers.

5. Model evaluation

5.1 Overview

The objective of this section is to test the possibility of incorporating the TCI into the software cost control tools, especially the EVM. This will be performed by testing whether the TCI correlates with one of the project cost variables. The evaluation process consists of the following activities.

1. Calculating the DCI of each developer as in **Equation 1**.
2. Calculating the TCI of the whole team as in **Equation 2**.
3. Calculating the planned and actual software development costs.
4. Calculate the cost of software reworks associated with defect removal as a result of the software tests.
5. Calculating the correlation coefficients between the TCI and the project cost variables.
6. Assessing whether the results show a strong correlation or not.
7. Assessing whether the TCI can be monitored and controlled during the project lifecycle.

The testing experiment is to be executed using multiple datasets. Each dataset represents a project containing the following data:

1. The competencies of the software developers as described by the SDCF.
2. The planned and actual development costs for the whole project.
3. The software rework data, which consists of:
   a. Number of developers worked in each round of software reworks
   b. Working days spent in each round of software reworks
4. Report status at the defined checkpoints, which consists of:
   a. The size of the developed software and the contribution of each developer, calculated in function points.
   b. The planned and actual cost for the software developed
   c. The planned and actual schedule for the software developer.

5.2 Data collection

We attempt to collect data from many sources, including the public data repositories, commercial datasets, relevant research groups, individual scholars, software firms, open source software communities, and relevant social media websites such as Research Gate. The data collection efforts have covered organizations and individuals scattered in 11 countries. Only three software firms agreed to share their data. The first firm agreed to share the project data of three financial applications, the second one shared the data of its healthcare application, and the third one shared its administrative application. However, the data collection attempts are shown in **Table 3**.

In fact, we face many challenges in collecting the data due to the following reasons:
1. The public data repositories and the commercial datasets, concerned with software engineering and project management, do not include the required data.
2. Software firms are reluctant to share their cost-related data. This type of data is mostly classified as confidential.
3. The software firms that agreed to share their data do not adopt the FPA as part of their software development methodologies.
4. Some of the competencies of the SDCF are not found in the performance appraisal procedures as defined by the software firms.
5. Parts of the project data are missing due to poor project tracking and poor documentation.

To fill these the gaps, we took the following actions:
1. We had 8 interviews with the project managers and team leaders to capture the missing data, especially the technical details related to the assessment of the developed software at each checkpoint.
2. We reviewed some minutes of meetings to extract missing milestone dates.
3. In some projects, defect removal activities are not well-documented. We extracted the required data by searching the correspondences among the developers, project managers and testing teams.
4. We submitted the team competency forms, shown in Table 1 and **Table 2**, to be filled by the project managers, team leaders, and human resource personnel. Moreover, we facilitated workshops to assist them to fill the forms in a collective manner.

On the other hand, to conduct the FPA, we took the following actions:

Data Source	Contacted Sources	Response			Shared Projects	
		None	Negative	Positive	Irrelevant	Relevant
Public data repositories	4		Open access		4	0
Commercial datasets	1	0	0	1	1	0
Relevant research groups	3	2	0	1	1	0
Software firms	31	13	15	3	1	5
Individual scholars	13	5	8	0	0	0
Open source communities	3	3	0	0	0	0
Questions posted in Research Gate	Public Post	0	0	0	0	0

TABLE 3: Data collection attempts

1. We reviewed the requirement documentations, software architectures, screens, and database schemas. This is to assess the size and complexity of the software.
2. We conducted 5 FPA workshops accompanied with presentations and 3 focus groups to convert the raw data to the variables used by the TCI model.

5.3 Data processing

To calculate the DCIs and TCIs for the whole collected data, we established a well-defined set of procedures as shown in **Table 4**.

TABLE 7: %R per project

Project ID	% R
P01	11.4%
P02	16.3%
P03	11.5%
P04	12.0%
P05	15.6%

TABLE 4: Data processing procedure

#	Tasks	Used Technique
1	Calculate the DCI for all the developers per each project	Review the developer competency forms
2	Conduct FPA: Define the system functions per project	1. Review the requirements documentation and application architecture 2. Review the application screens and database alternatively
3	Conduct FPA: a. Assign weights and determining complexity for each system function b. Calculate the software size	1. Review the requirements documentation and application architecture 2. Review the application screens and database alternatively 3. FPA workshops
4	Calculate the planned costs in person-days	1. Review project plans to identify the planned schedule and assigned developers 2. Interview with the development team and project managers
5	a. Map the system functions to the project schedule b. Identify the actual durations and assigned developers c. Calculate the actual cost (AC) in (person-days) d. Calculate the developed FPs per developer e. Calculate % complete (%C), f. %C = (Developed FPs/ total software size) %	Review the status reports at checkpoints

Table 5 shows the software sizes, the contributions of the individual developers, planned and actual costs, and the costs associated with reworks and defects removal. Moreover, we assessed the developers by assigning values to each criterion. We applied **Equation 2** to calculate the individual DCIs. Then, we used **Equations 2 and 3** to calculate the TCI for each project as shown in **Table 6**. We assumed equal weights for all competencies. **Table C.2** in **Appendix C** shows the DCI calculations in detail.
We have also calculated the percent cost of rework (%R) using **Equation 5**. The results for each project is shown in **Table 7**.

$$\%R = \frac{CoR}{CoR+CoD} X\ 100 \qquad (5)$$

Where CoR and CoD are the cost of rework and cost of development respectively.

Project ID	Size (FPs)	Cost of Development (person-day)			Developed FPs/Developer			
		Planned	Actual	Rework	D_1	D_2	D_3	D_4
P01	632	392	537	69	170	151	143	46
P02	369	262	297	58	88	112	109	0
P03	631	445	491	64	208	164	172	0
P04	256	180	206	28	141	115	0	0
P05	294	294	357	66	155	139	0	0

TABLE 5: Projects' aggregate data

Project ID	Size (FPs)	Cost of Development (person-day)			Developed FPs/Developer			
		Planned	Actual	Rework	D_1	D_2	D_3	D_4
P01	632	392	537	69	170	151	143	46
P02	369	262	297	58	88	112	109	0
P03	631	445	491	64	208	164	172	0
P04	256	180	206	28	141	115	0	0
P05	294	294	357	66	155	139	0	0

TABLE 6: DCIs and TCIs values

5.4 Correlation test

We tested the correlation of the TCI with the corresponding values of the planned, actual, rework costs, and the percent rework cost, yielding the results shown in **Table 8**.

Correlated Variable	Correlation Factor
Planned Cost	0.14
Actual Cost	0.05
Rework Cost	-0.59
%R	-0.79

TABLE 8: TCIs correlation results

A correlation factor of ±0.8 or more indicates a strong correlation, ±0.5 indicates moderate correlation, while ±0.2 or less indicates weak or no correlation (Zou et al., 2003). The results in **Table 8** show no correlation between the TCI and both the planned and actual development costs as the correlation factors, in both cases, are less than 0.2. Moreover, there is a moderate correlation between the TCI and the rework cost. On the other hand, there is somehow a strong negative correlation of -0.79 between the TCI and the %R. This correlation can be interpreted as follows: a team with low TCI is expected to deliver a poor-quality product with a high percentage of the cost of rework while a team with high TCI is expected to deliver a good quality product with a low percentage of the cost of rework.

5.5 TCI monitoring and controlling

To illustrate how the team performance could be monitored, **Figure 2** shows the TCI of the five projects versus the percent complete (%C). The value of %C is calculated by dividing the software size developed at a specific time by the total planned size of the software. The data associated with the diagram represent the measured data at each checkpoint during the project's lifecycle. **Table C.3** in **Appendix C** shows these data.

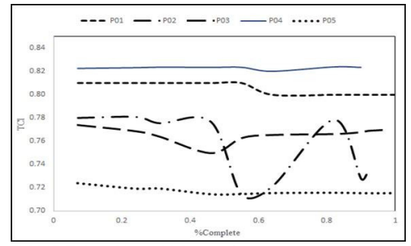

FIGURE 2. TCI vs. %Complete

In general, to maximize the TCI, the developers with the highest DCIs should have the highest workloads i.e. the highest IFs. On the other hand, developers who have the lowest DCIs should have the lowest IFs. Clearly, this should be governed by the project constraints such as time and cost.
The following hypothetical example illustrates the use of the TCI to monitor and control the team performance in a project.

Example

Suppose that there is a team consists of two developers with DCIs of 0.9 and 0.7.

It is planned that the first developer has a share of 40% of the total assignment while the second one has 60%.

Hence, DCI1 = 0.9, DCI2 = 0.7, IF1 = 0.4, and IF2 = 0.6
TCI = 0.9 x 0.4 + 0.7 x 0.6 = 0.78

If the value of TCI is not accepted to the project management team and there is a need to increase the TCI. This increase could be achieved by three scenarios:
The first one is to replace the second developer by a one with a higher DCI (0.85 for example) while keeping the same impact factors. Hence, TCI = 0.9 x 0.4 + 0.85 x 0.6 = 0.87
The second scenario is to increase the impact factors of the developers whose DCIs are high. In this case, the workloads could be adjusted to increase the share of the first developer (to 0.6 for example). The IF of the second developer would be reduced to 0.4. Hence, TCI = 0.9 x 0.6 + 0.7 x 0.4 = 0.82
The third scenario implies changing the DCI and the IF. For example, the second developer is replaced (as in the first scenario) and the IFs are adjusted (as in the second scenario), yielding: TCI = 0.9 x 0.6 + 0.85 x 0.4 = 0.88
Therefore, the overall TCI can be modified by reconfiguring the values of the DCIs or/and IFs.

6. Results and Discussion

6.1 Results

This study establishes a unified framework for software development competency; namely the SDCF. The SDCF covers the technical and nontechnical competencies in a well-structured manner (Answer to RQ1). Providing a scale for rating the individual competencies enables the construction of a model quantifying the competency of the individual developers and the team containing them; namely the DCI and TCI (Answer to RQ2). Moreover, the model evaluation shows a strong correlation between the TCI and the cost of software rework expressed as a percentage of the total cost. This finding indicates the possibility of incorporating the TCI into the software cost control models by enabling these models to forecast the software reworks and to monitor and control the performance of the individual developers and their teams (Answers to RQ3 and RQ4). In general, this study has scientific and practical implications as will be discussed in the subsequent subsections.

6.2 Scientific implications

Our literature review has indicated a huge knowledge gap in studies quantifying human factors in software projects. This supports the arguments of Mishra and Mishra (2011) and Lenberg et al. (2015) that most of the research and practice has focused on the technical factors without satisfying the needs for more studies related nontechnical issues. The available studies have only "scratched the surface of their impacts" on the software development projects (Capretz et al., 2017). In the light of the gap in this kind of studies, we believe that this research forms a strong contribution. It establishes a new framework, which extends and integrate the SWEBOS and SWEBOK. It also provides a quantified model extending the framework developed by Colomo-Palacios et al. (2013). Moreover, this study opens new avenues for researching the human side of the software development in a quantitative manner. In fact, there is a need for more empirical studies in the human-related issues (Mishra and Mishra, 2011). On the other hand, this research brings the attention to the relationship between the TCI and the cost of the software reworks and show the possibility of future researches to develop a software cost control model incorporating the human factors.

6.3 Practical implications

We believe that this study can assist the software firms, project managers, and software team leaders in many aspects. The SDCF can be used to adjust the performance appraisal so that the competencies measured could be in harmony with these used in the software development projects. Moreover, it encourages the software project managers to consider the human factors while managing projects. Project managers and team leaders may use the DCI and TCI in planning, monitoring, and controlling the performance of the development team. They may also make use of the relationship between the TCI and the cost of rework to minimize the total cost of development.

6.4 Threats to validity

The threats to the validity of this research consist of threats to internal validity, construct validity, and external validity. Our discussion in this section will follow the guidelines suggested by Kitchenham et al. (2002).

6.4.1 Internal validity

In this study, the TCI calculation is an average of the DCIs of the individual developers. Nevertheless, teams are not a simple collection of individuals. The current TCI does not reflect the performance of the team in terms of collective attributes such as homogeneity, level of diversity, coherence, synergy, and trust. Moreover, the maximum number of individuals in the sample teams does not exceed four persons. Therefore, the impacts of the team size on the collective attributes, team performance, and the individual performance are not investigated.
The collected data did not include agile projects. Agile methodologies, such as Scrum, have more focus on the human side. Values such as commitment, courage, focus, openness and respect are embodied and practiced by the Scrum team members (Schwaber and Sutherland, 2017). These values may affect the individual and collective attributes of the developers. The absence of investigating all these factors may impose threats to the internal validity of this study.

6.4.2 Construct validity

The software development experience influences function point estimates (Ho-Leung, 2005). Accordingly, we expect that the varied experience of the developers participated to the measurements may affect the accuracy of counting the function points. On the other hand, the subjectivity, skills, and experiences of the project managers and team leaders who filled the developers' competency forms may have a strong influence on the accuracy of both the DCI and TCI calculations.

6.4.3 External validity

Although the TCI model is successfully evaluated, we suggest that our model is neither comprehensive, nor generic. The following threats may impact its

- Due to the mentioned difficulties in data collection, the number of the datasets used in evaluating the TCI model is very few.
- We did not test projects with large sizes greater than 1000 function points. The large-size projects may consist of different interacting teams and may have more complex factors that are not investigated by this study.
- We did not test projects with a large size of team members such as 10 or 20 developers.
- The TCI values in the sample ranged from 0.72 to 0.82. Therefore, wide ranges of values below 0.7 or exceeding 0.82 are not tested.
- We did not test projects adopting methodologies other than the waterfall projects.

7. Conclusion and future work

This research establishes a unified software competency framework consisting of the technical and nontechnical competencies needed by the software developers. Moreover, it introduces the DCI and TCI indices to quantify the individual developer's competencies as well as the collective competency of the development team.
As a step towards incorporating the human factors in the software cost control models, this research shows a correlation between the team competency and the percent of the cost of software reworks, which is not provided by the current software cost control models such as the EVM. Moreover, the suggested team competency index itself could be used as a monitoring and controlling tool throughout the lifecycle of the project. All these contributions may have both scientific and practical benefits.
The direction of the future research is to extend the EVM by incorporating human factors such as the TCI. Also, the impacts of the collective attributes of the software teams need deeper investigation. Moreover, the impact of the software development lifecycles, especially the agile methods, will be studied.

Appendix A. EVM in brief

The Project Management Body of Knowledge (PMBOK) describes the EVM main parameters as follows:

- · Budget At Complete (BAC): is the authorized budget for the whole project activities.
- · Planned Value (PV): is the authorized budget assigned to the work scheduled.
- · Actual Costs (AC): is the amount of money spent for the work accomplished. In fact, it is the sunk cost.

- Earned Value (EV): is the percent of the total budget completed at a point in time. EV is usually calculated by multiplying the budget for an activity by the percent progress for that activity. EV is usually calculated as follows.

$$EV = \%C \times BAC \quad \text{(A.1)}$$

Where %C is known as the Percent Complete of the work planned.

In terms of the parameters mentioned above, EVM provides two indicators to measure the schedule and cost performances. These are:

- Schedule Performance Index (SPI): It measures how the project team is efficiently performing their tasks. It is calculated as follow.

$$SPI = \frac{EV}{PV}. \quad \text{(A.2)}$$

- Cost Performance Index (CPI): Provides a measure of the cost efficiency of the budgeted resources. It is given by:

$$CPI = \frac{EV}{AC}. \quad \text{(A.3)}$$

CPI < 1 indicates that there is a cost overrun while SPI > indicates that the actual performance is ahead of the estimated schedule. CPI = 1 indicates that the cost efficiency is as planned.

Then, EVM provides forecasting of the project cost according to the current performance. This is done through the Estimate AT Complete (EAC). EAC is given by:

$$EAC = \frac{BAC}{CPI}. \quad \text{(A.4)}$$

$$EAC = AC + \frac{BAC - EV}{SPI \; X \; CPI}. \quad \text{(A.5)}$$

Equation A.4 is used when focusing on the cost performance, while **Equation A.5** is used when considering the impact of both the cost and schedule performances.

Appendix B. FPA in brief

FPA measures the functional requirements of the software according to their complexities. It classifies software into five distinct system functions, mainly; external inputs, outputs, inquiries, external interfaces to other systems, and logical internal files. Each system function has its weight according to the level of complexity as shown in **Table B.1** (IFPUG, 2010).

Function type	Complexity		
	Low	Average	High
Internal Logical File	7	10	15
External Interface File	5	7	10
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6

TABLE B.1: FPA system functions and their associated sizes

To use **Table B.1**, a detailed analysis of each of the system functions is performed in terms of the following factors:

- The number of data element types (DET)
- The number of record element types (RET)
- The number of file types referenced (FTR)

The sum of the weights of all system function determines the unadjusted size of the software in function points (UFP). This value is adjusted by a multiplier called the Value Adjustment Factor (VAF). The VAF is determined depending on the complexity of the system characteristics in the operational environment. The weights are shown in Table B.2.

GSC	Global System Characteristics	Weight (0-5)
1	Data Communication	
2	Distributed Data Processing	
3	Performance	
4	Heavily Used Configuration	
5	Transaction Rate	
6	Online Data Entry	
7	End User Efficiency	
8	Online Update	
9	Complex Processing	
10	Reusability	
11	Installation Ease	
12	Operational Ease	
13	Multiple Sites	
14	Facilitate Change	
Total		

TABLE B.2. Global system characteristics

The weights are assigned according to the degree of complexity of each characteristic. Then, the Value Adjustment Factor (VAF) is calculated as follows:

VAF = 0.65 +0.01 x Total

Finally, the adjusted function points are given by,

FP = VAF x UFP

Appendix C. Projects' data

Project ID	Round 1		Round 2		Round 3		CoR (person-day)
	Days	Developers	Days	Developers	Days	Developers	
P1	24	2	9	2	3	1	69
P2	19	2	8	2	2	2	58
P3	16	2	10	2	6	2	64
P4	12	2	4	1	-	-	28
P5	22	2	12	2	2	1	66

TABLE C.1: Cost of reworks (CoR)

Com. ID*	Crit. ID**	P01				P02			P03				P04		P05	
		D_1	D_2	D_3	D_4	D_1	D_2	D_3	D_1	D_2	D_3	D_4	D_1	D_2	D_1	D_2
C_1	C_{11}	5	5	4	4	5	4	4	5	5	4	4	5	4	4	4
	C_{12}	4	4	4	3	4	4	4	4	4	4	3	5	4	4	3
	C_{13}	4	4	4	4	4	4	3	4	4	4	4	4	4	4	3
	C_{14}	5	4	4	3	4	4	3	4	4	4	3	5	4	4	3
C_2	C_{21}	5	5	4	3	5	4	4	5	5	4	3	5	4	4	4
	C_{22}	5	5	4	3	5	4	4	5	5	4	3	5	4	3	3
	C_{23}	5	3	3	3	4	3	3	4	3	3	3	4	3	3	3
	C_{24}	5	4	4	2	4	4	4	5	4	4	2	5	4	3	3
C_3	C_{31}	5	4	4	4	4	4	4	5	4	4	4	5	4	4	4
	C_{32}	5	4	4	4	4	4	4	5	4	4	4	5	4	4	4
	C_{33}	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4
C_4	C_{41}	5	4	4	4	4	4	4	4	4	4	4	4	4	4	3
	C_{42}	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3
	C_{43}	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3
C_5	C_{51}	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
	C_{52}	4	3	3	3	4	4	3	4	3	3	3	4	3	4	3
C_6	C_{61}	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3
	C_{62}	4	3	4	4	4	4	4	4	3	4	4	4	4	3	3
	C_{63}	4	4	3	4	4	4	4	4	4	3	4	5	3	3	3
C_7	C_{71}	5	4	5	3	5	4	4	5	4	5	3	5	4	4	4
	C_{72}	5	3	4	4	5	4	4	5	3	4	4	5	3	4	3
	C_{73}	5	4	4	4	4	4	4	5	4	3	4	5	4	3	3
	C_{74}	5	4	4	4	4	4	3	5	4	3	4	5	4	3	3
C_8	C_{81}	5	2	4	4	3	3	3	3	2	4	3	3	4	5	4
	C_{82}	4	3	4	4	4	4	4	4	3	3	4	4	4	4	4
	C_{83}	4	3	3	3	4	3	3	4	3	3	3	4	3	3	4
	C_{84}	4	3	4	4	4	4	4	4	3	4	4	4	4	3	3
C_9	C_{91}	4	4	3	3	4	4	4	4	4	3	3	5	3	4	4
	C_{92}	4	4	3	3	4	4	4	4	4	3	3	4	3	4	4
	C_{93}	5	4	3	3	4	4	4	5	4	3	3	5	3	4	4
	C_{94}	5	3	3	3	4	4	3	5	3	3	4	5	3	4	3
C_{10}	C_{101}	5	3	4	3	4	4	3	5	3	3	4	5	4	4	4
	C_{102}	5	3	4	3	4	4	3	5	3	3	4	5	3	4	3
	C_{103}	4	3	4	3	4	3	3	4	3	3	4	4	3	4	3
Total		155	126	129	119	140	132	125	149	126	123	122	153	124	128	116
DCI		0.91	0.74	0.76	0.70	0.82	0.78	0.74	0.88	0.74	0.72	0.72	0.90	0.73	0.75	0.68

TABLE C.2. Developers' competency analysis

*Com. ID = Competency ID, **Crit. ID =Criteria ID

Project ID	%C	TCI
P01	7	0.81
P01	24	0.81
P01	31	0.81
P01	46	0.81
P01	55	0.81
P01	64	0.80
P01	82	0.80
P01	90	0.80
P01	92	0.80
P01	96	0.80
P01	100	0.80
P02	16	0.78
P02	31	0.78
P02	46	0.78
P02	64	0.78
P02	74	0.71
P02	79	0.72
P02	86	0.78
P02	94	0.73
P02	100	0.73
P03	07	0.77
P03	11	0.77
P03	19	0.76
P03	29	0.75
P03	32	0.76
P03	44	0.77
P03	56	0.77
P03	74	0.77
P03	89	0.77
P03	94	0.77
P04	10	0.82
P04	30	0.82
P04	37	0.82
P04	49	0.82
P04	71	0.82
P04	80	0.82
P04	90	0.82
P04	100	0.82
P05	06	0.72
P05	12	0.72
P05	18	0.72
P05	21	0.71
P05	28	0.71
P05	34	0.72
P05	40	0.72
P05	45	0.72
P05	49	0.72
P05	56	0.72
P05	59	0.72
P05	67	0.72
P05	70	0.72
P05	77	0.72
P05	84	0.72
P05	89	0.72
P05	94	0.72
P05	100	0.72

TABLE C.3: Calculations of the TCI vs. %C

References

Acuña, S. T., Gómez, M. N., Hannay, J. E., Juristo, N., Pfahl, D. (2015). Are team personality and climate related to satisfaction and software quality? Aggregating results from a twice replicated experiment. Information and Software Technology, 57, 141-156.

AlQaisi, R., & Gray, E. (2013). Echoes from the Field: An Empirical Study of Contemporary Software Engineering Practices in Some Software Development Organizations in the UK. In BCS Quality Specialist Group 21st Annual SQM 2013 Conference, BCS London, UK (Vol. 4).

AlQaisi, R., Gray, E., Steves, B. (2017). Software systems engineering: A journey to contemporary agile and beyond, do people matter?. In BCS Achieving Software Quality in Development and Use. SQM, BCS, Southampton, UK, , pp. 159-173

AlQaisi, R., Gray, E., Moffat, D., Wang, B. (2016). 'Echoes from the Field'Study Outcome and Discussion: Reflections on Software Systems Engineering Practice. In INCOSE International Symposium (Vol. 26, No. 1, pp. 1624-1638).

Anderson, G., Keith, M. J., Francisco, J., Fox, S. (2018). The Effect of Software Team Personality Composition on Learning and Performance: Making the" Dream" Team. In Proceedings of the 51st Hawaii International Conference on System Sciences.

Bennatan, E. M. (2000). On time within budget: software project management practices and techniques. John Wiley & Sons, Inc.

Bloch, M., Blumberg, S., Laartz, J. (2012). Delivering large-scale IT projects on time, on budget, and on value. Harvard Business Review. [Online]. Available: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value. [Accessed: 10-May-2017].

Boehm, B. W. (1981). Software engineering economics (Vol. 197). Englewood Cliffs (NJ): Prentice-hall.

Bourque, P. and Fairley, R.E. (eds.)(2014). Guide to the software engineering body of knowledge, Version 3.0, IEEE Computer Society; www.swebok.org.

Broza, G. (2012). The Human Side of Agile: How to help your team deliver. 3P Vantage Media, ISBN-13: 978-0988001626

Capretz, L. F., Ahmed, F., Silva, F. Q. (2017). Soft Sides of Software. Journal of Information and Software Technology, 92(2017), 92-94.

Casanovas, J., Colom, J. M., Morlán, I., Pont, A., Ribera, M. (2004). Libro Blanco sobre las titulaciones universitarias en Informática en España [White Book: University degrees in computer engineering]. Madrid: ANECA.

Chandrasekaran, S., Gudlavalleti, S., Kaniyar, S. (2014). Achieving success in large complex software projects. McKinsey & Company, 1-5. [Online]. Available: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/achieving-success-in-large-complex-software-projects. [Accessed: 10-June-2015].

Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., GarcíA-PeñAlvo, F. J., Tovar-Caro, E. (2013). Competence gaps in software personnel: A multi-organizational study. Computers in Human Behavior, 29(2), 456-461.

Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., García-Peñalvo, F. J., Tovar, E. (2014). Project managers in global software development teams: a study of the effects on productivity and performance. Software Quality Journal, 22(1), 3-19.

de Souza, A. D., Rocha, A. R. C., Cristina, D., Constantino, B. A. (2014, June). A Proposal for the Improvement of Project's Cost Predictability Using Earned Value Management and Quality Data–An Empirical Study. In European Conference on Software Process Improvement (pp. 170-181). Springer, Berlin, Heidelberg.

DeMarco, T., Lister, T. (2013). Peopleware: productive projects and teams. Upper Saddle River, NJ: Addison-Wesley.

Ghosh, S. (2015). Systemic Comparison of the Application of EVM in Traditional and Agile Software Project. Integration, 5, 3. [Online]. Available: http:// pm.umd.edu/files/public/documents/student-papers/2011/ [Accessed: 18/5/2015)]

International Function Points Users Groups (IFPUG) (2010). Function Point Counting Practices Manual Release 4.3.1. Netherland.

Ju, H., Xu, S. (2017). Research Status of Earned Value Management. In Proceedings of the Fourth International Forum on Decision Sciences (pp. 449-459). Springer, Singapore.

Ho-Leung, T. S. O. I. (2005). To evaluate the function point analysis: a case study. International Journal of the Computer, the Internet and Management, 13(1), 31-40.

Khalid, T. A., Yeoh, E. T. (2015, September). Controlling software cost using fuzzy Quality based EVM. In Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE), 2015 International Conference on (pp. 275-280). IEEE.

Lenberg, P., Feldt, R., Wallgren, L. G. (2015). Behavioral software engineering: A definition and systematic literature review. Journal of Systems and software, 107, 15-37.

Manifesto, Chaos (2013). Think big, act small. The Standish Group International Inc, 176. [Online]. [Accessed: 22-Dec-2016].

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K., Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. IEEE Transactions on software engineering, 28(8), 721-734.

Meyer, A. N., Barton, L. E., Murphy, G. C., Zimmermann, T., Fritz, T. (2017a). The work life of developers: Activities, switches and perceived productivity. IEEE Transactions on Software Engineering, 43(12), 1178-1193.

Meyer, A. N., Zimmermann, T., Fritz, T. (2017b). Characterizing Software Developers by Perceptions of Productivity. In Empirical Software Engineering and Measurement (ESEM), 2017 ACM/IEEE International Symposium on (pp. 105-110). IEEE.

Mishra, D., Mishra, A. (2011). A review of non-technical issues in global software development. International Journal of Computer Applications in Technology, 40(3), 216-224.

Naeni, L. M., Shadrokh, S., Salehipour, A. (2014). A fuzzy approach for the earned value management. International journal of project management, 32(4), 709-716.

Naval Air Forces Organization (NAVAIR) (2004), Using software metrics & measurements for earned value toolkit [Online]. Available: https://acc.dau.mil/adl/en- [Accessed: 10/5/2015]

Pracharasniyom, K., Utsugi, S., Koizumi, Y., Hirose, S., Aso, H., Konosu, T. (2015) Human Resource Management in Small-scale Project. Journal of Business Administration and Languages [Online]. Available: http://journal.tni.ac.th/upload/files/pdf/Human%20Resource%20Management%20in%20Small-scale%20Project.pdf [Accessed: 10/10/2017]

Project Management Institute, PMI (2017). A guide to the project management body of knowledge – 6th ed. Newton Square, PA: PMI.

Schwaber, K, Sutherland, J (2017). The scrum guide. The definitive guide to scrum: The rules of the game. Scrum.org, 268.

Sedelmaier, Y., Landes, D. (2014). Software engineering body of skills (SWEBOS). In Global Engineering Education Conference (EDUCON), 2014 IEEE (pp. 395-401). IEEE.

Shneiderman, B. (1980). Software psychology: Human factors in computer and information systems. Cambridge, MA: Winthrop Publishers

Solomon, P. J. (2005, July). 1.4. 2 Performance-Based Earned Value®. In INCOSE International Symposium (Vol. 15, No. 1, pp. 180-197).

Vargas, R. V. (2004). Using earned value management indexes as a team development factor and a compensation tool. In Prague: Project Management Institute Global Congress EMEA.

Waychal, P., Capretz, L. F. (2017). Need for a Soft Dimension. arXiv preprint arXiv:1704.00801.

Xu, J., Zhang, H., Li, F. (2010, December). Project integrated management based on quality earned value. In Information Science and Engineering (ICISE), 2010 2nd International Conference on (pp. 432-435). IEEE.

Yilmaz, M., O'Connor, R. V., Colomo-Palacios, R., Clarke, P. (2017). An examination of personality traits and how they impact on software development teams. Information and Software Technology, 86, 101-122.

Zou, K. H., Tuncali, K., Silverman, S. G. (2003). Correlation and simple linear regression. Radiology, 227(3), 617-628.

AUTHORS

Tarig Ahmed Khalid

Tarig Ahmed Khalid is a PhD research scholar at Multimedia University, Malaysia. He obtained his B.Sc. and M.Sc. in electrical engineering from University of Khartoum, Sudan. He is a certified project management professional (PMP). His research interests include software project management, requirements engineering, and fuzzy systems. He is a senior member of the IEEE and member of the PMI.

Eng-Thiam Yeoh

Eng-Thiam Yeoh is a Senior Lecturer in Faculty of Computing & Informatics, Multimedia University, Malaysia. He obtained his PhD from Multimedia University in 2009, after obtaining his M.Phil degree from University of Cambridge, England (1991) and his first degree in Computer Science from National University of Malaysia (1989). His research interests include e-learning, software engineering, multimedia education, fuzzy systems, natural language processing and big data. He is a member of the IEEE