Empirical Validation of Requirements Management Measures

Mahmoud Khraiwesh[1] and Asim El Sheikh[2]
[1]Faculty of Science and Information Technology, Zarqa Private University, Jordan
[2]Faculty of Information System, Arab Academy for Banking and Financial Sciences, Jordan

Abstract: *Requirements management measures help organizations to understand, control and assess requirements management process. This paper validates empirically a set of requirements management measures. The measures were defined for the five specific practices of requirement's management key process area in capability maturity model integration by applying the goal question metrics paradigm to the five specific practices. We have applied the defined measures on three information systems using historical data. Then, for each information system some hypotheses have been followed to confirm the validity of the defined measures empirically.*

Keywords: *Requirements management, measures, measures validation, CMMI, GQM.*

Received August 19, 2007; accepted December 9, 2007

1. Introduction

The only way to improve any process is to measure specific attributes of the process, develop a set of meaningful metrics based on these attributes, and then use the metrics to provide indicators that will lead to strategy for improvement. Software measurement plays an important role in understanding and controlling software development practices and products [11]. Measurement is a mechanism for characterizing, evaluating, and predicting for various software processes and products [2].

Requirements are the foundation of the software development process. Carefully developed software requirements are a key issue for project success [10]. Since requirements often change, even during development, it is important to control the continuing definition of requirements as they change throughout the software life cycle to be able to anticipate and respond to requests of change [18]. Our rationale for concentrating on this early phase of the software process was that problems in this area have a profound effect on system development costs and functionality [22].

Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to characterize the attributes by clearly defined rules (and scales) [7]. Measurement is important for three basic activities: understanding, control and improvement [6]. Reasons for measuring are: to assess achievement of quality goals, to determine status with respect to plans, to gain understanding of processes, products, resources, and environments, to establish baselines for comparisons with future assessments and track improvement efforts [15] .

Software measurement is currently in a phase in which terminology; principles and methods are still being defined and consolidated. We should not expect to find quantitative laws that are generally valid and applicable, and have the same precision and accuracy as the laws of Physics, for instance. As a consequence, the identification of universally valid and applicable measures may be an ideal, long term research goal, which cannot be achieved in the near future [5]. Software engineering is not grounded in the basic quantitative laws of physics. Direct measure such as voltage, mass, velocity, or temperature, are uncommon in the software world. Because software measures and metrics are often indirect, they are open to debate [17].

Lamsweerde [12] conducted a survey of over 8000 projects from 350 US companies and revealed that one third of the projects were never completed and one half succeeded only partially, that is, with partial functionalities, major cost overruns, and significant delays. When asked about the causes of such failures, executive managers identified poor requirements as the major source of problems (about half of the responses) - more specifically, the lack of user involvement (13%), requirements incompleteness (12%), changing requirements (11%), unrealistic expectations (6%), and unclear objectives (5%). On the European side, a recent survey of over 3800 organizations in 17 countries similarly concluded that most of the perceived software problems are in the area of requirements specification (greater than 50%) and requirements management (50%).

Hall *et al.* [9] carried out a case study of 12 companies at different levels of capability as measured by the CMM. They discovered that, out of a total of 268 development problems cited, almost 50% (128) were requirements problems. Organizations from industry, government, and the Software Engineering Institute (SEI) joined together to develop the CMMI Framework, a set of integrated CMMI models.Two kinds of materials are contained in the CMMI model [1]:

- Materials to evaluate the contents of the processes-information that is essential to technical, support and managerial activities.
- Materials to improve process performance-information that is used to increase the capability of the organization's activities.

The Goal Question Metric (GQM) paradigm to process and metrics was developed by Basili and Weiss [3] as a technique for identifying meaningful measures for any part of the software process. It has proven to be a particularly effective approach to selecting and implementing measures.

In our previous work [10] we analyzed the five specific practices defined in the requirements management Key Process Area (KPA) of the CMMI [19]. By means of the GQM paradigm [2], we defined nearly 70 measures.

This paper validates empirically the defined measures in [10] for the five specific practices of requirements management KPA in CMMI-SW (staged representation) model and confirms that they really measure the five specific practices. The five specific practices are: obtain an understanding of requirements, obtain commitment to requirements, manage requirements changes, maintain bidirectional traceability of requirements, and identify inconsistencies between project work and requirements.

The remainder of the paper is organized as follows. Section 2 describes measurement theory. Section 3 describes the related work on measures validation. Section 4 describes the validity of the defined measures empirically. Finally, section 5 presents conclusions and future research.

2. Measurement Theory

Software measurement is concerned with deriving a numeric value for an attribute of a software product or process. By comparing these values to each other and to standards that apply across an organization, one may be able to draw conclusions about the quality of software or software process.

Measurement is not solely the domain of professional. We use it in every day life. Price acts as a measure of values of an item in a shop. When making a journey, we calculate distance, choose our route, measure our speed, and predict when we will arrive at our destination. So measurement helps us to understand our world, interact with our surroundings and improve our lives.

Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to defined rules [6, 20]. Software measurement is concerned with deriving a numeric value for some attributes of product or process. By comparing these values to each other and to standards that apply in the organization we can conclude the quality of the product or process [21].

Measurement captures information about attributes of entities. An entity is an object (such as person or a room) or an event (such as a journey) in real world. We want to describe the entity by identifying characteristics that are important to distinguishing one entity from another. An attribute is a feature or property of an entity, the area or color of a room and the cost of a journey. When we describe entities by using attributes, we often define the attributes using numbers or symbols. Some software engineers claim that important attributes like dependability, quality, usability and maintainability are simply not quantifiable; we prefer to try to use measurement to advance our understanding of them [6].

Formally, we define measurement as mapping from empirical world to formal, relational world. Consequently, a measure is the number or symbol assigned to an entity in order to characterize an attribute by this mapping. We begin in the real world, studying the entity. Thus the real world is the domain of the mapping and the mathematical world is the range.

The purpose of performing the mapping is to be able to manipulate data in the numerical system and use the result to draw conclusions about the attribute in the empirical system. We refer to our measuring mapping as a measurement scale. We classify measurement scales as one of the five major types: nominal, ordinal, interval, ratio, and absolute [6].

Although some companies have introduced measurement programs, most organizations still don't make systematic use of software measurement. Because the software processes are poorly defined and controlled, and are not sufficiently mature to make use of measurements. Another reason is that there are few established standards in this area.

Software metrics may be either predictor metrics used to predict product attributes or control metrics used to control the software process [21]. In software there are three classes of entities and attributes we wish to measure.

- Processes: are collections of software-related activities.

- Products: are any artifacts, deliverables or documents that result from a process activity.
- Resources: are entities required by a process activity (example: documentation from previous phase).

Within each class of entity, we can distinguish between internal and external attributes.

- Internal attributes of a product, process or resources: are those that can be measured purely in terms of the product, process or resources itself.
- External attributes of a product, process or resources: are those that can be measured only with respect to how the product, process or resources relates to its environment.

It is impossible to measure software quality attributes directly. Quality attributes such as maintainability, understandability and usability are external attributes that relate to how developers and users see the software. They are affected by many factors and there is no simple way to measure them [21]. The relationship between the internal and the external attributes should be clear and validated. (Example, stability of requirements is an external attribute, while number of requirements changes is internal attributes).

Direct measurement of an attribute of an entity involves no other attributes or entity (length of source code measured by lines of code, duration of testing process measured by elapsed time in hours). Indirect measurement of an attribute of an entity involves other attributes or entity.

3. Measurement Validation

Each attribute in the empirical system corresponds by the measurement to an element in a number system, so that by studying the numbers, we learn about the real world. Thus, we want the mapping to preserve the relation. This rule is called the representation condition, because the measure represents the attribute in the numerical world. The representation condition asserts that a measurement mapping *M* must map entities into numbers and empirical relations into numerical relations. Any measure that satisfies the representation condition is a valid measure. We can say that our intuition about the way the world works is the starting point for all measures. [6]. Measurement validation means that measures must represent accurately those attributes they claim to quantify [11]. If *X* is taller than *Y* our observation reflects that X is taller than *Y*, we can say that "taller than" is the empirical relation and the numerical relation is >. When we say X height is 91 centimeters and *Y* height is 71 centimeters, we really mean that we are measuring height by mapping each person into centimeters. We can say that *X* is taller than *Y* if and only if M(X) > M(Y), which is satisfied because in the numerical world M(X) = 91 and M(Y) = 71, representation condition is satisfied. Suppose that in the numerical world M(X) = 70 and M(Y) = 90, the representation condition is not satisfied, because it does not correspond to what happen in the real world.

Several definitions of measures validations are present in the literature. The most recognized is the internal-external validation. Fenton and Pfleeger [6] define measure validation as: validating a software measure is the process of ensuring that the measure is a proper numerical characterization of the claimed attribute by showing that the representation condition is satisfied. This definition is also known as internal validation, which is validation in a narrow sense.

Theoretical validation involves the intuitive understanding of the attribute we want to measure [4]. Researchers assume that the theoretical validation is not sufficient; they expect that the measure is part of the prediction system [6]. No theoretical model can guarantee the validity of a measure, the measure must be validated empirically [4].

External validation is done by showing that an external attribute is function of an internal one. We prove that an external attribute *X* verifies the equation X=f(Y) where *Y* is an internal attribute [13].

The same concepts of internal and external validation are used in the definition of theoretical and empirical validation [11]. Theoretical validation allows us to say whether a measure is valid with respect to some defined properties as the list defined by Fenton and Pfleeger. External attributes are mostly indirect measures and internal attributes are mostly direct measures.

When we perform an empirical validation we verify that measured values of attributes are consistent with values predicted by models involving the attribute [11]. Empirical validation is based on the proof that internal attributes are connected to external attributes. In other words, with empirical validation we prove that a measure is useful, i.e., that it is connected to a goal [4]. Connection between internal and external attributes can only be determined empirically [6]. The measures which are defined for external attributes can only be validated empirically [14]. No requirements management measures have been validated, and only few empirical studies have been performed in the area of RM measures [14].

The goal or the relation can be expressed as a hypothesis and then test the hypothesis to see if the data we collect will confirm or refute the hypothesis we have stated [6]. A hypothesis proposes a specific relationship between a measure and some useful attribute or the supposition which explains the behavior you want to explore; we must conduct an experiment to test the hypothesis. Whenever possible, we should state the hypothesis in quantifiable terms, so that it is easy to test the hypothesis. A hypothesis

captures our intuitive understanding of the studied phenomena [5].

The entity which is related to our work is the requirements management process. This entity can have several attributes, the ones we would like to measure are understandability, commitment, manage requirements, traceability, and consistency, which determine whether the general goals of the requirements management KPA are reached. These are external attributes and the measures defined for these attributes can only be validated empirically.

3.1. Fuzziness in Measures Validation

It is not possible to theoretically validate a measure without performing an empirical study, because the representation condition can only be proven empirically [6]. Theoretical validation of the measure is often not possible and a large number of measures have never been subject to an empirical validation [5].

The proof that the representation condition is satisfied can only be empirical by its nature [16, 24]. The validity of measures connected to many external attributes can only be performed by external validation [13]. Empirical validation requires a large number of data and it is not possible for some measures [14].

The empirical validation would require a large amount of data and rarely be conducted in the proper way. Empirical validation of some measures is not possible. Few measures have been validated because there is no widely accepted way of validating measures [23, 16, 24, 14]. This situation has frequently led to some degree of fuzziness in the measures validation [5]. Measures validation is extremely difficult and one should not expect a single researcher to provide, within one study, a complete and definitive validation [4]. We should not expect to find quantitative laws that are generally valid and applicable, and have the same precisions and accuracy as the laws of Physics [5, 7].

4. Validity and Reliability of the Defined Measures

We have applied the defined measures on three information systems. The information systems are in the maintenance phase, so we have entered the historical data for the last 6 months of two information systems and the historical data for the last 18 months of one information system.

The experiment has been performed in the computer centre at Zarqa Private University (Jordan) on three information systems: human resource information system, continuous teaching information system, and library information system. For each information system we have described the collected data for all the defined measures. Then, for each information system some hypotheses have been followed to show the validity of the defined measures empirically.

4.1. The Human Resource System

The human resource application is a medium system, it is a production system. It has 14 main requirements as a baseline and 1 new requirement. The total number of items related to all the requirements is 156 items (form, report, table …*etc*). We have followed the changes to the requirements for a period of six months and tracked all the affected items because of the change.

Following are some hypotheses:

- Hypothesis: 100 % of the requirements providers must have direct relation to work, which is important for the understanding of requirements. Collected data. The 15 requirements providers of the 15 requirements have relation to work. No misunderstood requirements, no missing requirements, and no rejected requirements. So requirements providers' relation to work is a good predictor for understanding.
- Hypothesis: each requirement must be elicited from more than one level of requirements providers, which is important for the understanding of requirements.
- Collected data: there are 2 levels of requirements providers for each of the 15 requirements. No misunderstanding in requirements, no missing requirements, and no rejected requirements. So the number of levels the requirements providers are from is a good predictor for understanding.
- Hypothesis: 100 % of the requirements must have shared understanding between the requirements providers and the practitioners, which is important for the understanding of requirements.
- Collected data: the 15 requirements providers have shared understanding with practitioners about the 15 requirements. No misunderstanding in requirements, no missing requirements, and no rejected requirements. So requirements providers' relation to work is a good predictor for understanding.
- Hypothesis: 80% of the requirements changes are implemented and delivered within the estimated time, which is important for the commitment to requirements.
- Collected data: From 65 requirements changes proposed, 63 are approved, 1 is rejected, 1 is To Be Determined (TBD).

The 63 approved requirements changes are implemented, 3 under testing and 60 requirements changes are delivered.

From 65 requirements changes, 59 are processed within the estimated time. (More than 80% of requests to change were processed within the estimated time which means the commitment was obtained).

- Hypothesis: the number of change requests for each requirement must be identified, which is important to manage requirements changes.
- Collected data: 5 requirements have more than 5 requests to change (one requirement has 31 requests to change; one requirement has 15 requests to change). 10 requirements have less than 5 requests to change (4 requirements have 0 request to change).
- Hypothesis: acurrent statuses of all requests to change must be identified, which is important to manage requirements changes.
- Collected data: the total number of requests to change is 65 requests. From them, 60 requests to change were delivered, 3 requests to change are under testing, 1 request to change was rejected, and 1 request to change is TBD.
- Hypothesis: all project items that are affected by the request to change must be identified, which is important to manage requirements changes.
- Collected data: the total number of items affected by all the changes is 121 changes to items. From the 121 changes to items, 64 were reports, 33 were forms, 20 were tables, 2 were menus, and 2 were software media. The number of items that were changed for the 15 requirements are: 3, 41, 6, 35, 5, 4, 1, 16, 6, 0, 0, 0, 0, 2, 2 respectively. The number of items affected for each request to change was identified.
- Hypothesis: source of all requests to change must be identified, which is important to maintain traceability.
- Collected data: the source of each of the 65 of requests to change is identified. The source is the department and the practitioner who issue the request to change.
- Hypothesis: for each inconsistency case the source of inconsistency (report, form, database …*etc*) and the reason of inconsistency (requirement provider, practitioner...) must be identified, which is important to identify inconsistency between the requirement and the product.
- Collected data: there is no inconsistency case.

4.2. The Continuous Teaching System

The continuous teaching is a medium system, it is a production system. It has 5 main requirements as a baseline. The total number of items related to all the requirements is 91 items (form, report, table …). We have followed the changes to the requirements for a period of 18 months and tracked all the affected items because of the change.

Following are some hypotheses

- Hypothesis: 100 % of the requirements providers must have direct relation to work, which is important for the understanding of requirements.
- Collected data: the 5 requirements providers of the 5 requirements have relation to work. No misunderstanding in requirements, no missing requirements, and no rejected requirements. So, requirements providers' relation to work is a good predictor for understanding.
- Hypothesis: each requirement must be elicited from more than one level of requirements providers, which is important for the understanding of requirements.
- Collected data: there are 2 levels of requirements providers for each of the 5 requirements. No misunderstanding in requirements, no missing requirements, and no rejected requirements. So the number of levels the requirements providers are from is a good predictor for understanding.
- Hypothesis: 100 % of the requirements must have shared understanding between the requirements providers and the practitioners, which is important for the understanding of requirements.
- Collected data: the 5 requirements providers have shared understanding with practitioners about the 5 requirements. No misunderstanding in requirements, no missing requirements, and no rejected requirements. So requirements providers' relation to work is a good predictor for understanding.
- Hypothesis: 80% of the requirements changes are implemented and delivered within the estimated time, which is important for the commitment to requirements.
- Collected data: from 22 requirements changes proposed, 21 are approved, 1 is rejected. The 21 approved requirements changes are implemented and delivered. From 22 requirements changes 20 are processed within the estimated time. More than 80% of request to change were processed within the estimated time which means the commitment was obtained.
- Hypothesis: the number of change requests for each requirement must be identified, which is important to manage requirements changes.
- Collected data: 1 requirement has more than 5 requests to change (13 requests to change). 4 requirements have less than 5 requests to change (4 requirements have 0 request to change).
- Hypothesis: all current statuses of all requests to change must be identified, which is important to manage requirements changes.
- Collected data: the total number of requests to change is 22 requests. From them, 21 requests to change are delivered, and 1 request to change is rejected.

- Hypothesis: all project items that are affected by the request to change must be identified, which is important to manage requirements changes.
- Collected data: the total number of items affected by all the changes is 47 changes to items. From the 47 changes to items, 40 were reports, and 7 were forms. The number of items that were changed for the 5 requirements are: 5, 35, 0, 7, 0 respectively. The number of items affected for each request to change was identified.
- Hypothesis: source of all requests to change must be identified, which is important to maintain traceability.
- Collected data: the source of each of the total number 22 of requests to change is identified. The source is the department and the practitioner who issue the request to change.
- Hypothesis: for each inconsistency cases the source of inconsistency (report, form, database …*etc*) and the rationale of inconsistency (requirement provider, practitioner...*etc*) must be identified, which is important to identify inconsistency between the requirement and the product.
- Collected data: there is no inconsistency case.

4.3. The Library System

The library system is a medium system, it is a production system. It has 19 main requirements as a baseline and 1 new requirement. The total number of items related to all the requirements is 346 items (form, report, table …*etc*). We have followed the changes to the requirements for a period of six months and track all the affected items because of the change.

Following are some hypotheses:

- Hypothesis: 100 % of the requirements providers must have direct relation to work, which is important for the understanding of requirements.
- Collected data: the 19 requirements providers of the 19 requirements have relation to work. No misunderstanding requirements, no missing requirements, and no rejected requirements. So requirements providers' relation to work is a good predictor for understanding.
- Hypothesis: each requirement must be elicited from more than one level of requirements providers, which is important for the understanding of requirements.
- Collected data: there are 2 levels of requirements providers for each of the 19 requirements. No misunderstanding requirements, no missing requirements, and no rejected requirements. So the number of levels the requirements providers are from is a good predictor for understanding.
- Hypothesis: 100 % of the requirements must have shared understanding between the requirements providers and the practitioners, which is important for the understanding of requirements.
- Collected data: the 19 requirements providers have shared understanding with practitioners about the 19 requirements. No misunderstanding in requirements, no missing requirements, and no rejected requirements. So requirements providers' relation to work is a good predictor for understanding.
- Hypothesis: 80% of the requirements changes are implemented and delivered within the estimated time, which is important for the commitments to requirements.
- Collected data: from 58 requirements changes proposed, 54 are approved, 1 is analyzed, 1 is rejected, and 2 is TBD. The 54 approved requirements changes are delivered. From 58 requirements changes 56 are processed within the estimated time. More than 80% of request to change were processed within the estimated time which means the commitment was obtained.
- Hypothesis: the number of change requests for each requirement must be identified, which is important to manage requirements changes.
- Collected data**:** 4 requirements have more than 5 requests to change (one requirement has 17 requests to change; one requirement has 10 requests to change...). 15 requirements have less than 5 requests to change (7 requirements have 0 request to change).
- Hypothesis: all current statuses of all requests to change must be identified, which is important to manage requirements changes.
- Collected data: the total number of requests to change is 58 requests. From them, 54 requests to change were delivered, 1 request to change is analyzed, 1 request to change was rejected, and 2 requests to change is TBD.
- Hypothesis: all project items that are affected by the request to change must be identified, which is important to manage requirements changes.
- Collected data: the total number of items affected by all the changes is 389 changes to items. From the 389 changes to items, 205 were reports, 172 were forms, 11 were tables, and 1 was menus. The number of items that were changed for the 19 requirements are: 0, 3, 0, 0, 3, 0, 38, 13, 6, 0, 2, 0, 21, 2, 118, 0, 0, 1, 182 respectively. The number of items affected for each request to change was identified.
- Hypothesis: source of all requests to change must be identified, which is important to maintain traceability.
- Collected data: the source of each of the total number 58 of requests to change is identified. The source is the department and the practitioner who issue the request to change.

- Hypothesis: for each inconsistency case the source of inconsistency (report, form, database …*etc*) and the rationale of inconsistency (requirement provider, practitioner...*etc*) must be identified, which is important to identify inconsistency between the requirement and the product.
- Collected data: there is no inconsistency case.

5. Conclusion and Future Research

This paper has proved empirically the validity of the defined measures in [10] by using historical data of three information systems. The information systems are in the production phase. For each information system we have described the collected data for all the defined measures. Then, for each information system some hypotheses have been followed empirically and proved that the collected data confirm the hypotheses.

The defined measures should be implemented on a number of large long-term projects. Another important area of future research is the definition of measures for other key process areas in CMMI.

References

[1] Ahern D., Clouse A., and Turner R., *CMMI Distilled: A Practical Introduction to Integrated Process Improvement*, Addison Wesley, 2003.

[2] Basili R. and Rombach D., "The TAME Project: Towards Improvement Oriented Software Environments," *Computer Journal of IEEE Transactions on Software Engineering*, vol. 14, no. 6, pp. 758-773, 1988.

[3] Basili R. and Weiss D., "A Methodology for Collecting Valid Software Engineering Data," *Computer Journal of IEEE Transactions on Software Engineering*, vol. 10, no. 6, pp.728-738, 1984.

[4] Briand C., El Eman K., and Morasca S., "Theoretical and Empirical Validation of Software Product Measures," *Technical Report*, Pennsylvania State University, 1995.

[5] Briand C., Morasca S., and Basili R., "An Operational Process for Goal Driven Definition of Measures," *Computer Journal of IEEE Transactions on Software Engineering*, vol. 28, no. 12, pp. 194-1200, 2002.

[6] Fenton N., Whitty R., and Yoshinori I., *Software Quality Assurance and Measurement: A Worldwide Perspective*, Elsevier Science, 1995.

[7] Fenton E. and Pfleeger L., *Software Metrics: A Rigorous and Practical Approach*, International Thomson Publishing, Boston, 1995.

[8] George D. and Mallery P., *SPSS for Windows Step by Step: A Simple Guide and Reference*, New Jersey, 2003.

[9] Hall T., Beecham S., and Rainer A., "Requirements Problems in Twelve Software Companies: An Empirical Analysis," *IEE Proceedings: Software*, USA, pp. 153-160, 2002.

[10] Khraiwesh M. and El Sheikh A., "Requirements Management Measures," *in Proceedings of the International Arab Conference on Information Technology (ACIT'2005)*, Jordan, pp. 124-128, 2005.

[11] Kitchenham B., Pfleeger L., and Fenton N., "Towards a Framework for Software Measurement Validation," *Computer Journal of IEEE Transactions on Software Engineering*, vol. 21, no. 12, pp. 23-37, 1995.

[12] Lamsweerde A., "Requirements Engineering in the Year: A Research Perspective," *in Proceedings of the 22nd International Conference on Software Engineering (ICSE'2000),* Ireland, pp. 5-19, 2002.

[13] Loconsole A., "Empirical Studies on Requirement Management Activities," *in Proceedings of 26st IEEE/ACM International Conference on Software Engineering*, UK, pp. 431-437, 2004.

[14] Loconsole A. and Brstler J., "Theoretical Validation and Case Study of Requirements Management Measures," *Internal Report*, Umea University, 2003.

[15] Park E., Goethert W., and Florac A., *Goal Driven Software Measurement*: *A Guidebook, Software Engineering Institute Handbook*, Carnegie Mellon University, 1996.

[16] Pfleeger L., Jeffery R., Curtis B., and Kitchenham B., "Status Report on Software Measurement," *Computer Journal of IEEE Software*, vol. 14, no. 2, pp. 179-192, 1997.

[17] Pressman S., *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 2005.

[18] Reifer J., "Requirements Management: The Search for Nirvana," *in Proceedings of IEEE Software*, China, pp. 45-47, 2000.

[19] Sarse K. and Jasper V., Software Engineering Institute, *Maturity Model Integrated Software Engineering Stage Representation*, Carnegie Mellon University, 2002.

[20] Solingen R. and Berghout E., *The Goal Question Metric Method: A Practical Guide for Quality Improvement of Software Development*, McGRAW-Hill, London, 1999.

[21] Sommerville I., *Software Engineering*, Addison Wesley, 2004.

[22] Sommerville I. and Ranson J., "An Empirical Study of Industrial Requirements Engineering Proces Assessment and Improvement," *Computer Journal of ACM Transactions on Software Engineering and Methodology*, vol. 14, no. 1, pp. 85-117, 2005.

[23] Weyuker E., "Evaluating Software Complexity Measures," *Computer Journal of IEEE*

Transactions on Software Engineering, vol. 14, no. 9, pp. 1357-1365, 1988.

[24] Zuse H., *A Framework of Software Measurement*, Walter de Gruyter, 1998.

Mahmoud Khraiwesh is an assistant professor at Faculty of Science and Information Technology in Zarqa Private University, Jordan. He received his MSc degree in computer science from Jordan University, Jordan, in 2002, and his PhD degree in computer information systems from the Arab Academy for Banking and Financial Sciences, Jordan, in 2006. His area of research is in requirements management measures.

Asim El Sheikh received his BSc (honors) from University of Khartoum, Sudan, an MSc and a PhD from University of UK. Professor El Sheikh worked for University of Khartoum, Philadelphia University in Jordan, and the Arab Academy for Banking and Financial Sciences, Jordan. He is currently the dean of the Faculty of Information System and Technology at the Arab academy for Banking and Financial Sciences. His areas of research interest are computer simulation and software engineering.