IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4123

A Comparative Analysis of Three Knowledge Areas of the Guide to the Software Engineering Body of Knowledge with the Rational Unified Process®

Michel Brouillette and Pierre Bourque
École de technologie supérieure, 1100 Notre-Dame Street West, Montreal, Canada, H3C 1K3
Tel: 514 396-8623, Fax: 514 396-8684, m_broue@hotmail.com, pbourque@ele.etsmtl.ca.

ABSTRACT

The goal of this paper is to highlight differences identified between three Knowledge Areas or chapters of the Guide to the Software Engineering Body of Knowledge and the Rational Unified Process. The differences identified are of two types: terminology differences and topics that can only be found in one or other of the compared documents. The paper shows that the two documents are quite consistent with regard to these types of differences for the Knowledge Areas considered.

INTRODUCTION

This paper reports on the findings of the comparative analysis of three Knowledge Areas of the Guide to the Software Engineering Body of Knowledge (SWEBOK) [1] and the Rational Unified Process (RUP®, version 2001.03.00.23) [2]. The three Knowledge Areas compared are software requirements, software testing and software configuration management.

The goal of this paper is to highlight differences identified between these three Knowledge Areas or chapters of the Guide to the SWEBOK and the RUP. The differences identified are of two types: terminology differences and topics that can only be found in one or other of the compared documents.

GUIDE TO THE SOFTWARE ENGINEERING BODY OF KNOWLEDGE (SWEBOK)

The Guide to the SWEBOK project is an initiative of the IEEE Computer Society, and has the support of many public and private organizations. The purpose of the project is to build a consensus on the core body of knowledge of software engineering. Articulating a body of knowledge is an essential step toward recognition of software engineering as a profession because it represents a broad consensus on the content of the discipline.

The topics included in the Guide to the SWEBOK are deemed to be generally accepted and are grouped into the following Knowledge Areas:

- Software requirements;
- Software design;
- Software construction;
- Software testing;
- Software maintenance;
- Software configuration management;
- Software engineering management;
- Software engineering process;
- Software engineering tools and methods;
- Software quality.

RATIONAL UNIFIED PROCESS (RUP)

The Rational Unified Process (RUP) is a well-known and widely commercialized software engineering process description. The goal of RUP is to ensure the production of high-quality software which meets the needs of its end-users, within a predictable schedule and budget.

The RUP is based on the concept of software best practices. The best practices on which the RUP is built are the following:

1. Develop iteratively;
2. Manage requirements;
3. Use component architecture;
4. Model visually (UML);
5. Continuously verify quality;
6. Control change.

COMMON VIEWPOINT

Even though the objectives of the Guide to the SWEBOK and of the RUP are different, the concepts of best practice and generally accepted knowledge share many common elements (see definitions in Table 1).

Table 1: Definition of generally accepted knowledge and software best practices

SWEBOK - Generally Accepted Knowledge	RUP - Software Best Practices
"Generally accepted" means that the knowledge and practices described are applicable to most projects most of the time, and that there is widespread consensus about their value and usefulness. "Generally accepted" does not mean that the knowledge and practices described are or should be applied uniformly on all projects; the project management team is always responsible for determining what is appropriate for any given project. AND "Knowledge" is defined as what is to be included in the study material of a software engineer licensing exam that a graduate would pass after completing four years of work experience.	"Best practices" are commercially proven approaches to software development which, when used in combination, strike at the root causes of software development problems. They are "best practices" not so much because you can precisely quantify their value, but rather because they are commonly used in industry by successful organizations.

This paper appears in *Issues and Trends of Information Technology Management in Contemporary Organizations,* the proceedings of the Information Resources Management Association International Conference.

The definitions of generally accepted knowledge and software best practices share many commonalities. Notably, the decision on whether or not a topic is deemed to be generally accepted or a practice a "best practice" is not based on formal rules, it is rather a subjective decision based on the acceptance of the practice or knowledge topic. The two definitions emphasize broad usage or wide applicability. As well, both documents can be tailored to specific needs.

5 SUMMARY OF FINDINGS FOR SOFTWARE REQUIREMENTS

This knowledge area is concerned with the acquisition, analysis, specification, validation and management of software requirements. The RUP incorporates all these aspects. The Guide to the SWEBOK uses the term "requirements engineering" to describe the systematic handling of requirements. The RUP defines the systematic handling of requirements as requirements management. Requirements management in the RUP also provides the capability to establish and maintain agreement between the customer and the project team on the changing requirements of the system.

The Guide to the SWEBOK and the RUP have an equivalent definition of a requirement. The glossary of the RUP defines a requirement as a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification or other formally imposed document. The Guide to the SWEBOK defines a requirement as a property which must be exhibited in order to solve some real-world problem. The Guide to the SWEBOK states that requirements vary in intent and in the kind of properties they represent, and that a distinction can be drawn between product parameters and process parameters. The RUP makes no mention of process parameters, but identifies the two types of product parameters, functional requirements and non-functional requirements, which are also identified in the Guide to the SWEBOK. The Guide to the SWEBOK states that an essential property of all requirements is that they be verifiable; there is no such recommendation in the RUP. At most, the RUP states that all verifiable requirements should be tested, hence implicitly admitting that some requirements may not be verifiable.

The Guide to the SWEBOK states that eliciting stakeholder requirements is rarely easy and that the requirements engineer has to learn a range of techniques to help people articulate how they do their job and what might help them do their job better. The RUP suggests a number of such techniques.

The requirements engineer's job, as described in the Guide to the SWEBOK, differs from the role of requirements specifier described in the RUP. In addition to the responsibilities of the requirements specifier, the requirements engineer is also responsible for assessing the feasibility of the requirements.

The Guide to the SWEBOK states that, when a complete and coherent set of system requirements emerges from the analysis process and the system requirements are then allocated or distributed to subsystems/components, this is part of the architectural design. The RUP adopts a similar view.

The Guide to the SWEBOK states that it is almost always impractical to implement requirements engineering as a linear deterministic process where system requirements are elicited from stakeholders, baselined, allocated and handed over to the software development team. The RUP approach to requirements management maps perfectly to this affirmation.

Based on the comparative analysis, two topics have been identified as candidates for generally accepted knowledge to the Guide to the SWEBOK development team. These topics are the categorization of requirements and use-cases. Regarding RUP, differences from the Guide to the SWEBOK were identified for the following items: prioritization of requirements in the artifact Supplementary Specifications, requirements negotiation, verifiability of all requirements, and follow-up actions based on reported lessons learned.

SUMMARY OF FINDINGS FOR SOFTWARE TESTING

In the introduction to this chapter, the Guide to the SWEBOK mentions that the right attitude towards quality is one of prevention. It is therefore pointed out that software testing is an activity that should encompass the whole development process. This is exactly how testing is presented in the RUP. In fact, test planning is executed from the first phase or inception phase.

In the Guide to the SWEBOK, it is stated that testing is conducted in view of a specific test objective. In the RUP, the test objectives are stated in the artifact named Test Guidelines under Goal of Testing. The test adequacy criteria are referred to in the RUP, as are the test completion criteria in the artifact named Test Guidelines. These criteria should answer the question: how much testing is enough for achieving the stated objective? The test selection criteria of the RUP are found in the Test Plan in the Requirements for Test and Test Strategy sections. They answer the question: which test cases should be selected for achieving the stated objective? For performance testing, this question is answered in greater detail in the workload analysis document.

The Guide to the SWEBOK makes a distinction between measures which evaluate the thoroughness of the test set and measures which provide an evaluation of the program being tested. In the RUP, this distinction is also made. There are two key types of measures: coverage and quality. Coverage measures evaluate the thoroughness of the tests and quality measures evaluate the program being tested in terms of reliability, stability and performance. The RUP Test workflow contains all the activities of a test process as described in the Guide to the SWEBOK.

Two topics have been identified as candidates for generally accepted knowledge to the Guide to the SWEBOK development team. These topics are security and documentation testing. Regarding RUP, differences from the Guide to the SWEBOK were identified for the following items: definition of a default, test effectiveness/appropriateness, error guessing, mutation testing, random testing and suggestion of code-based techniques other than the decision-to-decision path test technique.

SUMMARY OF FINDINGS FOR SOFTWARE CONFIGURATION MANAGEMENT

The Guide to the SWEBOK defines configuration management as the discipline of identifying the configuration of a system at distinct points in time for the purposes of systematically controlling changes to the configuration and of maintaining the integrity and traceability of the configuration throughout the system life cycle. In the RUP, configuration management describes the product structure and identifies its constituent configuration items, which are treated as single versionable entities in the configuration management process. Configuration management deals with defining configurations, building and labeling, and collecting versioned artifacts into constituent sets and maintaining traceability between these versions.

Based on the comparative analysis, two topics have been identified as candidate generally accepted knowledge topics. These topics are the identification of problems solved by software configuration management and the change history of a software configuration item. In the case of RUP, differences from the Guide to the SWEBOK were found for the following items: special requirements from the customer concerning the software configuration management process, software configuration management requirements from external regulatory bodies, the levels of baselines, access control to the project repository, deviations and waivers from the software configuration management process, and the purpose of the physical configuration audit versus the purpose given in the Guide to the SWEBOK.

SUMMARY AND FUTURE STEPS

The paper shows that, for these Knowledge Areas at least, the two documents are quite consistent. Further work is necessary and is now under way to complete the comparative analysis for all ten Knowledge Areas of the Guide to the SWEBOK. Both development teams must now also consider the differences put forward by the comparative analysis. Other comparative analyses similar to the one described in this paper are also planned for the Guide to the SWEBOK, notably with software engineering process improvement models and assessment methods.

ACKNOWLEDGMENTS

We thank Rational Software and especially Philippe Kruchten for their support of this project.

REFERENCES

[1]Abran, A., Moore, J. W., Bourque, P., Dupuis, R., Tripp, L. (2001). *Guide to the Software Engineering Body of Knowledge Trial version 0.95* http://www.swebok.org/documents/ironman/Guide to the SWEBOK - Ironman Version 095.PDF.

[2]Kruchten, P., (2000). *The Rational Unified Process: An Introduction* (2nd ed.). Reading: Addison Wesley Longman.