Coverage Analysis and Improvement of the Role Definitions of the Bombardier Software Engineering Process

Pierre Bourque* pbourque@ele.etsmtl.ca 1-514 396-8623
Youssef Belkebir* belkebir_y@iquebec.com 1-514-668-1648
Claude Y Laporte* claporte@ele.etsmtl.ca 1-514 396-8956
Mikel Doucet** mikel.doucet@ca.transport.bombardier.com
Edgardo Palza V* edgardo.palza-vargas.1@etsmtl.ca

** École de technologie supérieure*
1100, rue Notre-Dame Ouest, Montréal Québec, Canada, H3C 1K3.

*** Bombardier Transportation*
1101, rue Parent, St-Bruno Québec, Canada, J3V 6E6

ABSTRACT

In this paper, we present the results of a project conducted to analyze the coverage of the roles within the Bombardier Engineering System (BES) Software Engineering (SWE) process definition and subsequently improve the role definitions. This is the common software engineering process definition of Bombardier's Transportation division. Role definitions were analyzed using three internationally recognized software engineering reference documents: the Rational Unified Process (RUP) ,IEEE/EIA Standard 12207.0-1996, and the Guide to the Software Engineering Body of Knowledge (SWEBOK Guide).

1 INTRODUCTION

The purpose of this paper is to present a project, described in greater detail in [4], conducted to analyze and improve the role definitions within the Bombardier Engineering System – Software Engineering (BES SWE) process definition. The BES SWE is the common software engineering process definition of Bombardier's Transportation division, in which each role definition specifies the purpose of the role, identifies the core responsibilities assumed by the role, and the hard and soft skills needed to perform the role.

Created in 1974 to provide subway wagons for the Montreal Transit Authority, Bombardier Transportation grew rapidly through acquisitions to become the world largest manufacturer of rail material for moving people. Recently, the company created a number of corporate Centres of Competence in various engineering specialties. The software engineering Centre of Competence, located just outside Montreal, is where this project was conducted.

An important portion of the software produced or integrated by Bombardier Transportation is "safety or mission critical software". An erratic behavior of the control system activating brakes on a train, for example, could have potentially catastrophic consequences. In this context, the definition and usage of mature software engineering processes and practices is imperative. Precise role definitions are also essential to ensure that responsibilities to be assumed and activities to be performed are well understood by everyone, as are the hard and soft skills necessary to perform each role. This is especially true when software engineering projects are conducted across many international sites.

The notion of the role is a core concept in the BES SWE, as it is in all software engineering process definitions, as shown in Figure 1 (excerpted from [13]). Roles perform activities that produce and consume artifacts. Roles are also responsible for artifacts. Of course, the same role may be performed in a given project by many people, and, conversely, one person may perform many roles. At the outset of the project, the BES SWE included 24 roles, as listed in Table 1, divided into four categories.

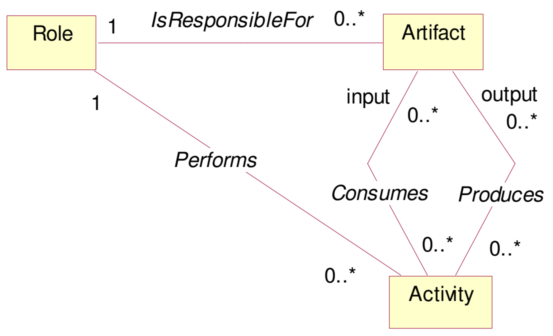

Figure 1 Interaction of roles, activities, and artifacts in a software engineering process definition

In order to facilitate the roll-out of the role definitions to all Bombardier Transportation software engineering sites, it was decided that the coverage analysis and subsequent improvements to the role definitions would be founded on internationally recognized reference documents. Notably, this was viewed as a way of adding credibility to the improved role definitions without giving the impression that one software engineering site was imposing its role definitions on the other sites. The selected reference documents were IEEE/EIA Standard 12207.0-1996, Standard for Information Technology — Software Life Cycle Processes [6], the Rational Unified Process (RUP) [8][1], and the Guide to the Software Engineering Body of Knowledge (SWEBOK Guide) [2][2]. It should be noted that role definitions are present as such in RUP. However, in the SWEBOK Guide and in IEEE/EIA Standard 12207.0-1996, information regarding the definition of roles is implied in the text, not often explicitly stated, and therefore had to be abstracted from the reference documents.

The three reference documents used to analyze and improve the BES SWE roles are presented in the next section. An example of how one specific role definition was analyzed and improved based on the SWEBOK Guide is then presented in the following section. A summary of findings and a conclusion are presented in the final section.

Table 1: BES SWE role definitions at the beginning of the project

Management Category	**Software Engineering – Supporting Category**
Senior Manager	Software Change Control Board
Project Manager	Software Infrastructure Administrator
Software Project Manager	Software Metrics Coordinator
Software Quality Assurance Manager	Software Process Engineer
Product Manager	Software Project Coordinator
Software Engineering Manager	Software Quality Assurance Engineer
Software Engineering Category	**Other categories**
Software Team Leader	Customer
Software Requirements Coordinator	Proposal Coordinator
Software Architect	Safety representative
Software Implementer	Software Trainer
Software Integrator	Software Training Coordinator
Software Test Designer	
Software Tester	

2 REFERENCE DOCUMENTS

This section briefly presents the three selected reference documents used to analyze and improve the role definitions in the BES SWE. They are the Rational Unified Process, IEEE/EIA Standard 12207.0-1996, and the SWEBOK Guide.

[1] Version 2001A.04.00 of RUP was used in this project.

[2] The Trial Version of the SWEBOK Guide published in 2001 was used in this project.

2.1 Rational Unified Process (RUP)

The RUP is a commercial object-oriented process framework for software development[3]. It includes a set of roles, activities, workflows, and artifacts which describe the *who*, the *how*, the *when* and the *what* of a software development process. It can be tailored to company specifics and to various sectors of the industry.

The RUP claims that it is based on industry-proven software best practices. These are:

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

The RUP is an iterative process described in two dimensions:

1- The time dimension: The horizontal axis in Figure 2 represents the life cycle or dynamic aspects of the process. These aspects are expressed in terms of cycles, phases, iterations, and milestones. In the RUP, a software product is designed and built in a succession of incremental iterations. This allows design ideas to be tested and validated, and risks to be mitigated, earlier in the life cycle.

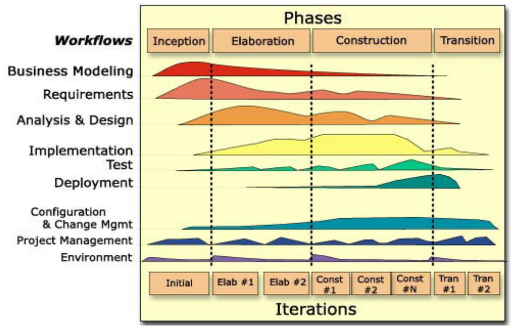

Figure 2. Overview of the RUP

2- The work dimension: The vertical axis in the figure represents the static aspect of the process. This axis represents core process disciplines (or workflows), which logically group software engineering activities by their nature. The static aspects comprise process components: activities, disciplines, artifacts, and roles. The RUP is composed of six core workflows, which are: business modeling, requirements, analysis and design, implementation, test, and deployment; and 3 supporting workflows, which are configuration management, project management, and environment.

2.2 IEEE/EIA Standard 12207.0-1996

The IEEE/EIA 12207.0-1996 Standard for Information Technology–Software Life Cycle Processes is considered a key standard regarding the definition of life cycle processes. It has notably been designated as the pivotal standard around which the Software Engineering Standards Committee (SESC) is harmonizing its entire collection of standards [11]. This standard groups software processes into activities and tasks, and these are organized into three categories: Primary Processes, which are divided into Acquisition, Supply, Development, Operation, and Maintenance; Supporting Processes, which are divided into Documentation, Configuration Management, Quality Assurance, Verification, Validation, Joint Review, Audit, and Problem Resolution; Organizational Life Cycle Processes, which are divided into Management, Infrastructure, Improvement, and Training.

2.3 The SWEBOK Guide

The objectives of the SWEBOK Guide are to characterize the content of the software engineering discipline, to promote a consistent view of software engineering worldwide, to clarify the place, and set the boundary, of software engineering with respect to other disciplines, and to provide a foundation for curriculum development and individual licensing material.

[3] See http://www-136.ibm.com/developerworks/rational/products/rup/

The SWEBOK Guide is a project of the IEEE Computer Society and has the support of numerous organizations. Subsequent to the publication of the Trial Version in 2001 [1] and a trial period, the 2004 Version of the SWEBOK Guide [2] was recently published. It is a consensually validated document available free of charge on www.swebok.org, and can also be purchased in book format from the IEEE Computer Society Press. The 2004 Version was approved by the IEEE Computer Society Board of Governors and will also be published as ISO Technical Report 19759.

The SWEBOK Guide is oriented toward a variety of audiences, all over the world. It is aimed at serving public and private organizations in need of a consistent view of software engineering for defining education and training requirements [5], classifying jobs, and developing performance evaluation policies and career paths [10]. It also addresses the needs of practising software engineers and software engineering managers, and the officials responsible for making public policy [3] regarding licensing and professional guidelines [12]. In addition, professional societies defining their certification rules, as well as educators drawing up accreditation policies for university curricula, will benefit from consulting the SWEBOK Guide, as will students of software engineering and educators and trainers engaged in defining curricula [7], [15] and course content [9].

The SWEBOK Guide seeks to identify and describe the subset of software engineering knowledge that is generally accepted. Generally accepted knowledge applies to most projects most of the time, and widespread consensus validates its value and effectiveness [14].

The SWEBOK Guide is subdivided into ten knowledge areas (KAs), the descriptions of which are designed to discriminate among the various important concepts, permitting readers to find their way quickly to subjects of interest. Upon finding such subjects, readers are referred to key papers or book chapters selected because they present the knowledge succinctly. The ten KAs are listed in Table 2. Each is treated as a chapter in the SWEBOK Guide.

Table 2: The SWEBOK Guide 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

3 COMPARING THE ROLE DEFINITIONS AND THE REFERENCE DOCUMENTS: AN EXAMPLE

In the course of this project, every role definition was individually analyzed against each of the three reference documents. An example of such an analysis for the Software Requirements Coordinator role using the SWEBOK Guide as the reference document is found in Table 3. Such a table was completed for each role definition against each of the three reference documents. The detailed improvements on the role definitions are based on these tables. The improved definition of the Software Requirements Coordinator role after the comparison with the three reference documents is found in Table 4 (proposed improvements are in italics).

Table 3 is composed of the following cells:

- Role name: name of the role in the BES SWE;
- OR: overall recommendation on the presence of the role resulting from the analysis with the reference document (Accept, Remove);
- GAP: overall evaluation of the difference between the definition of the role in the BES SWE and the definition of the role explicitly stated or implied in the reference document. Possible values are: Major, Minor, No Gap.
- RT: recommendation regarding the role title (Accept, Modify);
- P: recommendation regarding the role purpose section (Accept, Modify);
- CR: recommendation regarding the core responsibilities section (Accept, Modify);
- HS: recommendation regarding the hard skills section of the role definition (Accept, Modify);
- SS: recommendation regarding the soft skills section of the role definition (Accept, Modify).
- BES SWE: Excerpted text from the definition of the role in the BES SWE prior to improvement.

- SWEBOK: Excerpted text from the SWEBOK Guide relevant to this role definition and potentially useful for improving the definition of the role in the BES SWE.
- Note: Indications on how to improve the definition of the role based on this comparison.

Table 3: Analysis of the Software Requirements Coordinator Role using the SWEBOK Guide as the reference document

Role Name: Software Requirements Coordinator			**OR:** Accept		
GAP: Minor	**RT:** Modify	**P:** Modify	**CR:** Modify	**HS:** Modify	**SS:** Modify
BES SWE The Software Requirements Coordinator is responsible for requirements management of the overall project.					
SWEBOK (Chapter 2: Software Requirements) The Software Requirements KA is divided in the following manner: - Requirements Process includes Process Models, Process Actors, Process Support and Management, and Process Quality and Improvement. - Requirements Elicitation includes Requirement Sources and Elicitation Techniques. - Requirements Analysis includes Requirements Classification, Conceptual Modeling, Architectural Design and Requirements Allocation, and Requirements Negotiation. - Requirement Specification includes Requirements Definition Document, Software Requirements Specification (SRS), Document Structure and Standards, and Document Quality. - Requirements Validation includes Conduct of Requirements Reviews, Prototyping, Model Validation, and Acceptance Tests. - Requirements Managements includes Change Management, Requirement Attributes, and Requirements Tracing.					
Note The SWEBOK Guide uses the term *requirements engineer*rather than *software requirements coordinator*. The SWEBOK Guide is very useful for improving the hard skills needed for this role.					

Table 4: Definition of the Software Requirements Coordinator Role (proposed improvements are in *italics*)

Purpose: The Software Requirements Coordinator is responsible for requirements management of the overall software project. *More specifically, the software requirements coordinator is responsible for eliciting the requirements and establishing and maintaining an agreement with the customer on the requirements of the software project. The software requirements analyst analyzes, elaborates and refines the allocated requirements to ensure that they are feasible and appropriate to implement in software, clearly stated, consistent with one another, testable, and complete*.
Core Responsibilities: Responsible for the software requirements *engineering process, requirements elicitation, requirements analysis, requirements specification, requirements validation, and requirements management*. Responsible for requirements traceability and the generation of the Software Requirements Verification Traceability Matrix (SRVTM)
Hard Skills *Ability to implement software requirements engineering processes;* *Ability to acquire an understanding of the application and technology domain;* *Ability to elicit software requirements from system stakeholders and to overcome common obstacles to the elicitation process;* *Ability to describe mode and operating condition requirements;* *Ability to model software requirements using UML and CASE tools;* *Ability to analyze and negotiate software requirements;* *Ability to specify software requirements with selected documentation techniques;* *Ability to perform architectural design and requirements allocation;* *Ability to perform software requirements validation;* *Ability to perform software requirements change management;* *Ability to trace software requirements to software design artifacts;* *Ability to trace software requirements to test artifacts.* **Soft Skills:** *Ability to negotiate and resolve problems when conflicts occur;* *Active listening skills;* *Flexibility: Ability to adapt and deal with situations and manage expectations during periods of change;* *Sound business judgment: Knowledge of the business purpose of a project and decision-making within that context;* *Exhibition of several communication styles: Ability to recognize a person's communication style and adapt to it;* *Setting and managing of expectations;* *Ability to identify the key issues.*

Table 5: Consolidation of analyses and decision regarding the presence of the role

Role name (in alphabetical order)	RUP		IEEE 12207		SWEBOK		Global recom-mendation of the study	Decision regarding the presence of the role by Bombardier Transportation	Rationale for decision (when relevant)
	GAP	OR[4]	GAP	OR	GAP	OR			
Change Control Board	Minor	Accept	Minor	Accept	No	Accept	Accept	Accept	
Customer	Major	Accept	Minor	Accept	No	Accept	Accept	Accept	
Product Manager	Major	Remove	Minor	Accept	Minor	Accept	Accept	Accept	The Product Manager notably works closely with Business Development.
Project Manager	No	Accept	Minor	Accept	Minor	Accept	Accept	Accept	
Proposal Coordinator	Major	Remove	Minor	Accept	Major	Remove	Accept	Accept	Important role in the context of Bombardier Transportation
Safety Representative[5]	N/A	N/A	Major	Accept	N/A	N/A	Accept	Accept	Important role in the context of Bombardier Transportation
Senior Manager[6]	N/A	N/A	Minor	Accept	N/A	N/A	Accept	Accept	
Software Architect	No	Accept	Minor	Accept	Minor	Accept	Accept	Accept	
Software Engineering Manager	Major	Remove	Major	Remove	No	Accept	Accept	Accept	This role is implied in the CMM [16].
Software Implementer	No	Accept	Minor	Accept	Major	Accept	Accept	Accept	
Software Infrastructure Administrator	Minor	Accept	Minor	Accept	Minor	Accept	Accept	Accept	
Software Integrator	No	Accept	Minor	Accept	Minor	Accept	Accept	Accept	
Software Metrics Coordinator	Major	Remove	Major	Remove	Major	Accept	Accept	Accept	This role is notably important for process improvement.
Software Process Engineer	Minor	Accept	Minor	Accept	Minor	Accept	Accept	Accept	
Software Project Coordinator	Major	Remove	Major	Remove	Major	Remove	Remove	Remove	There is a major GAP in the three analyses.
Software Project Manager	Minor	Accept	Minor	Accept	Minor	Accept	Accept	Accept	
Software Quality Assurance Engineer	Major	Remove	Minor	Accept	Minor	Accept	Accept	Accept	
Software Quality Assurance Manager	Major	Remove	Minor	Accept	Minor	Accept	Accept	Accept	This role is notably implied in the CMM.
Software Requirements Coordinator	Minor	Accept	Minor	Accept	Minor	Accept	Accept	Accept	
Software Team Leader	Major	Remove	Major	Remove	Major	Remove	Remove	Accept	This role is implied in the

[4] Overall recommendation

[5] Safety functions are not covered in RUP and are out of scope of the SWEBOK Guide.

[6] Senior management functions are not within the scope of RUP and the SWEBOK Guide.

Role name (in alphabetical order)	RUP		IEEE 12207		SWEBOK		Global recom-mendation of the study	Decision regarding the presence of the role by Bombardier Transportation	Rationale for decision (when relevant)
	GAP	OR[4]	GAP	OR	GAP	OR			
									CMM.
Software Test Designer	No	Accept	Minor	Accept	Minor	Accept	Accept	Accept	
Software Tester	Minor	Accept	Minor	Accept	Minor	Accept	Accept	Accept	
Software Tool Specialist	No	Accept	Minor	Accept	Minor	Accept	Accept	Accept	
Software Trainer	Minor	Accept	Minor	Accept	Major	Remove	Accept	Accept	This role is implied in the CMM.
Software Training Coordinator	Minor	Accept	Minor	Accept	Major	Remove	Accept	Accept	This role is implied in the CMM.

4 SUMMARY OF FINDINGS AND CONCLUSION

In this paper, we have presented a project conducted to improve the software engineering role definitions within the software engineering process definition of a large multinational organization. The principal results of this project are consolidated in Table 5, including the final decision of Bombardier Transportation regarding the presence of each role in their software engineering process. In addition, detailed improvements were proposed to all role definitions and an illustration of these improvements was presented for the Software Requirements Coordinator role.

Well-defined roles notably ensure that everyone's responsibilities and the hard and soft skills necessary to perform each role are properly understood. This is especially important for critical software which is developed, integrated, and maintained across many international sites.

The detailed coverage analysis reported in this paper of the role definitions in the Bombardier System Engineering – Software Engineering (BES SWE) process definition and their subsequent improvements were based on a comparison with three internationally recognized documents. The selected approach was successful, and numerous improvements were brought to the role definitions at all levels of detail. Conducting a project similar to the one reported in this paper for roles included in the software engineering process definitions of other organizations could also yield useful results.

REFERENCES

[1] Abran, A., Moore, J.W., Bourque, P., and Dupuis, R. (eds.). *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society Press, 2001.

[2] Abran, A., Moore, J.W., Bourque, P., and Dupuis, R. (eds.). *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society Press, 2004.

[3] Aytaç, T., Ikiz, S. and Aykol, M., A SPICE-Oriented, SWEBOK Based, Software Process Based Assessment on a National Scale: Turkish Sector Software Survey – 2001. in *3rd International SPICE Conference*, (2003).

[4] Belkebir, Y. Analyse et amélioration des définitions de rôles du processus d'ingénierie logicielle du centre de compétence en génie logicielle de Bombardier Transport *Dept. of Electrical Engineering*, École de technologie supérieure, Montréal, 2003.

[5] Frailey, D. and Mason, J., Using SWEBOK for Education Programs in Industry and Academia. in *15th Conf. Software Engineering Education and Training Conference (CSEET 2002)*, (2002), 6-10.

[6] IEEE. IEEE/EIA 12207.0-1996 IEEE/EIA Standard Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology Software Life Cycle Processes, Institute of Electrical and Electronics Engineers, 1998.

[7] IEEE/ACM. Computing Curricula – Software Engineering Volume - Public Draft 3.1, Joint Task Force on Computing Curricula – IEEE Computer Society/Association for Computing Machinery, 2004.

[8] Kruchten, P. *The Rational Unified Process: An Introduction*. Addison-Wesley, 2003.

[9] Ludi, S. and Collofello, J., An Analysis of the Gap between the Knowledge and Skills Learned in Academic Software Engineering Course Projects and those Required in Real Projects. in *31st ASEE/IEEE Frontiers in Education Conference, 2001. http://fie.engrng.pitt.edu/fie2001/papers/1187.pdf*, (2001).

[10] McConnell, S. *Professional Software Development*. Addison-Wesley, 2004.

[11] Moore, J.W. *Software Engineering Standards – Aa User's Road Map*. Wiley-IEEE Computer Society Press, 1998.

[12] OIQ. Software Engineering Definitive Report, Ordre des ingénieurs du Québec, Montréal, Québec, 2000.

[13] OMG. Software Process Engineering Metamodel Specification, Object Management Group, 2002.

[14] PMI. A Guide to the Project Management Body of Knowledge, Project Management Institute, 2000.

[15] Ramakrishnan, S. and Cambrell, A., An In-forming Web-based Environment for a Bachelor of Software Engineering Degree – DoIT. in *IS2002 Informing Science + IT Education Conference*, (2002).

[16] SEI *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley, 1995.