Measuring the Cost of Software Quality of a Large Software Project at Bombardier Transportation: A Case Study

CLAUDE Y. LAPORTE
École de technologies supérieure

NABIL BERRHOUMA

MIKEL DOUCET
Bombardier Transportation

EDGARDO PALZA-VARGAS

In a competitive world, it is important to measure and improve the performance of software engineering processes and imperative to identify and eliminate rework that could have been avoided. The cost of software quality is one measure of the performance of software processes. It comprises the total cost associated with the prevention, appraisal, and correction of the anomalies in a product. A measure of the cost of software quality was performed by the software development group at Bombardier Transportation, a division of Bombardier Inc., located in Québec, Canada. A team of 15 software engineers developed the software to control the subway of a large American city. A project to measure the cost of software quality was carried out in four stages. A set of 27 classification rules was developed, and weights were assigned to each project task. More than 1100 software tasks were analyzed on a project totaling 88,000 hours. The measurements show that the cost of software quality represents 33 percent of the overall project cost. The cost of rework, or the cost of correcting anomalies, is 10 percent, the cost of prevention is 2 percent, and the cost of evaluation is 21 percent of the total development cost.

Key words

cost of anomalies, cost of evaluation, cost of prevention, cost of rework, cost of software quality, measurement, process improvement, very small entities

INTRODUCTION

According to Charette, "Studies have shown that software specialists spend about 40 to 50 percent of their time on avoidable rework rather than on what they call value-added work, which is basically work that's done right the first time. Once

a piece of software makes it into the field, the cost of fixing an error can be 100 times as high as it would have been during the development stage" (Charette 2005). Measuring and reducing the percentage of avoidable rework should be one objective of most process improvement initiatives.

A recent paper presented the results of a literature review about software quality cost research of 87 articles published between 1980 and 2009. One of the conclusions and recommendations was the following (Karg, Grottke, and Beckhaus 2011): Only about a third of the analyzed articles present a case study or more extensive empirical results. This appears to be insufficient for software quality cost research, which strongly relies on quantitative data to generate new findings. There is thus a need for novel approaches to gather quality cost data, as well as stronger cooperation between industry and research to make such data available. This article presents such a case study performed in the railway industry.

Many enterprises measure the costs required to perform various functions, such as the cost of developing a product, the cost of maintaining it, the cost of support, and so on. The measure of the cost of quality (CoQ) is very useful for improving the performance of processes, as one's objective must be to seek expensive activities and, above all, identify and eliminate waste. Importantly, software development activities should not be excluded from this measure. One important challenge is the measurement of the cost of software quality (CoSQ). It is clear that preventing defects reduces costs, and the CoQ is the sum of the costs associated with the prevention, assessment, and correction of anomalies. A framework for the classification of quality-related costs was introduced by Feigenbaum in the 1950s (Feigenbaum 1956) to provide justification for managers wishing to make investments in process improvement. The framework has since been adapted to the software engineering domain. In this article, the authors present the results of the measurement of the CoSQ of a large completed project.

OVERVIEW OF BOMBARDIER TRANSPORTATION

Bombardier Inc. has a workforce of some 80,000 people in 24 countries. Bombardier Aerospace is a world leader in the manufacture of business jets and regional transport, and Bombardier Transportation is a leader in the manufacture of rail transport equipment. The division manufactures locomotives, freight cars, and propulsion and control systems, and also provides systems and signaling equipment. Its product range includes passenger vehicles and transport systems.

Modern trains and subways are increasingly complex, and more and more subsystems are computer controlled, such as propulsion and braking systems. At the time of this case study, there were more than 30 software development centers within Bombardier Transportation, for a total of about 950 software engineers. One of these development centers is located in Québec, Canada.

The Software Development Group in Québec

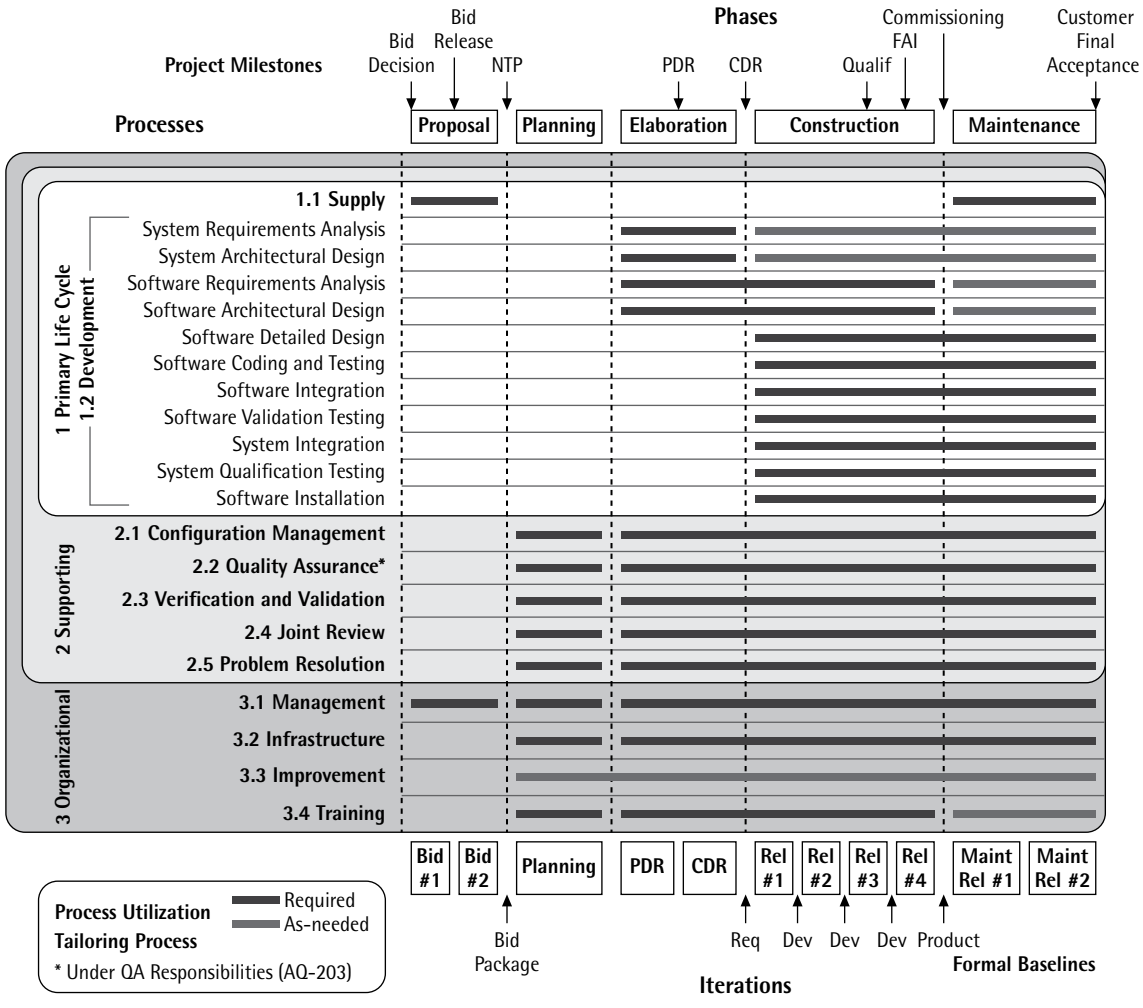
The Software Development Group (SDG), located in St-Bruno near Montreal, is made up of 30 software engineers whose role is to design, develop, and maintain embedded software for trains and subways, mainly the software monitoring system used for collecting software maintenance information and software for controlling car inclination.

Bombardier Transportation has defined, in their procedure, two types of objectives for the measurement of the CoQ: those directed at senior management and those targeting the SDG. These objectives are aligned with the main process of the ISO/IEC15939 standard (ISO 2007): commitment, planning, performing, and evaluating measurement activities for projects.

The objectives of a directive, directed at senior management, were the following:

- Quantify the CoQ components for high-level management
- Identify major opportunities to reduce costs
- Identify the main contributors to the cost of poor quality
- Provide a baseline to budget quality activities
- Stimulate improvement efforts through the publication of the CoQs within the company
- Develop a CoQ dashboard
- Use the measurements to compare process improvement activities, in order to identify those that are the most effective

FIGURE 1 The Bombardier software development process



Legend
NTP: Notice to proceed
Qualif: System qualification
Dev: Development
PDR: Preliminary design review
FAI: First article inspection
Rel: Release
CDR: Critical design review
Req: Requirement
Maint: Maintenance

The objectives targeting the SDG were:

- Identify a project to measure the CoSQ
- Collect data on costs
- Categorize costs related to software quality
- Develop a data model for the measurement of CoSQ
- Analyze data collected on the selected project using the data model
- Present the CoSQ report to senior management
- Expand the measurement of the CoSQ to all software projects in SDG

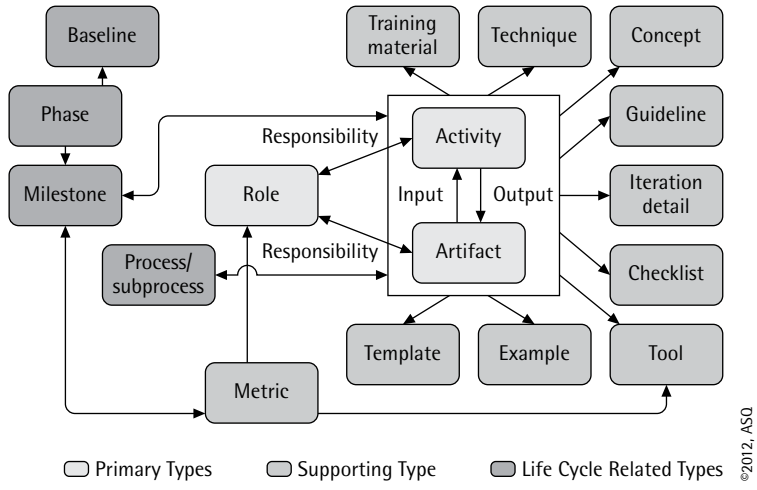
The Bombardier Software Development Process

The Bombardier Software Development Process (BSEP) was inspired by and derived mainly from the Rational Unified Process (RUP). Illustrated in Figure 1, it provides a disciplined approach to assigning tasks and responsibilities within a software development organization. Its goal is to ensure the production of high-quality software that meets the needs of end users within a predictable timeframe and budget.

The BSEP has two dimensions:

- The vertical axis represents processes that are groups of activities based on the ISO 12207 standard (ISO 2008). This dimension corresponds to the static aspect of the process, that is, how it is described in terms of process items: processes/subprocesses, activities, and artifacts.
- The horizontal axis represents time and shows the life-cycle aspects of the process as it unfolds. This dimension corresponds to the dynamic aspect of the process as it unfolds, and is expressed in terms of phases, iterations, milestones, and formal baselines.

FIGURE 2 Life-cycle-related types of the BSEP



The BSEP processes are expressed through a set of roles, activities, and artifacts:

- Roles: A role defines the behavior and responsibilities of an individual or groups of people working in teams, in the context of a software engineering organization. The roles and responsibilities define both the "who and how" the work will be executed. Individual members of project teams can play different roles during the project, wearing different hats, metaphorically speaking.
- Activities: Roles have activities assigned to them that define their work and that must be completed to achieve a given objective. An activity is a unit of work performed by an individual responsible for the activity described by the role. An activity is also any work performed by managers and technical staff to carry out project tasks. An activity is used to plan and monitor a project.
- Artifacts: An artifact is a product of the process and is the input or output of activities. Artifacts are used to perform activities and are produced during project execution. They may be internal or external to the project and take various forms:
  - A model, such as use-case model
  - A document, like the project plan, the requirements document, or software requirements specifications (SRS)
  - Code

Process Types

Process *types* constitute the foundation of the BSEP. Each process *item* that composes the BSEP has an associated type, and there are three of them (see Figure 2): a primary type, such as a role, activity, or artifact; a supporting type, such as a tool, training, or a metric; and a life-cycle type, such as a phase or a milestone. For example, a milestone is defined as a major event at a specific point in the project that has a significant bearing on the status of the work. It is an opportunity to check progress and reassess plans when management "Go" and "No Go" decisions are made. Milestones are located at the ends of phases or iterations.

BSEP Configuration Identification

The BSEP configuration identification (CI), also called the process bill of materials (BOM), is the cornerstone of the BSEP process. This document lists all of the process items that will form the final BSEP process. This approach is similar to the initial creation of a plan before the detail-writing effort for a new document is begun: identify the document structure and its elements, and provide high-level information for

TABLE 1 Configuration identification of the BSEP process

Primary life-cycle processes		
	Supply	
	Development	
		System requirements analysis
		System architectural design
		Software requirements analysis
		Software architectural design
		Software detailed design
		Software coding and testing
		Software integration
		Software validation testing
		System integration
		System qualification testing
		Software deployment
Supporting life-cycle processes		
	Configuration management	
	Quality assurance	
	Verification and validation	
		Peer reviews
		Verification evaluation
		Verification and validation reporting
	Joint review	
		Project management reviews
		Technical reviews
	Problem resolution	
		Problem report
Organizational life-cycle processes		
	Management	
		Estimate the project
		Plan the project
		Manage risks
		Track the project
	Infrastructure	
		Plan infrastructure activities
		Establishment of the project infrastructure
		Establishment of the global infrastructure
		Maintenance of the infrastructure
	Training	
		Plan training activities
		Implement training plan

those elements. The CI of BSEP provides similar information for the BSEP process:

- Process structure (based on the ISO 12207 software life-cycle processes standard)
- Process items or a list of process elements that will form the final BSEP process
- Associated types (for example, role, activity, artifact) for each process item
- Planning and monitoring information

Table 1 presents the configuration identification of the BSEP process.

COST OF SOFTWARE QUALITY

The Project Management Body of Knowledge (PMBOK®) from the Project Management Institute (PMI) defines CoQ as follows (PMI 2008): [Technique] a method of determining the costs incurred to ensure quality. Prevention and appraisal costs (cost of conformance) include costs for quality planning, quality control (QC), and quality assurance to ensure compliance to requirements (that is, training, QC systems, and so on). Failure costs (cost of nonconformance) include costs to rework products, components, or processes that are noncompliant, costs of warranty work and waste, and loss of reputation.

The concept of CoQ has been adapted for the software industry and is sometimes called the cost of software quality (CoSQ) (Campanella 1990; Mandeville 1990; Slaughter, Hanter, and Krishnan 1998; Krasner 1998; Houston 1999). CoSQ can be broken down into three categories: appraisal or evaluation costs, cost of prevention, and cost of anomalies (adapted from Galin 2004a and PMI 2008):

- Appraisal or evaluation costs: the costs of verification or evaluation of a product or service during the various stages of delivery (for example, reviews and tests).
- Cost of prevention: the costs incurred by an organization to prevent the occurrence of errors in various stages during the delivery process (for example, design, development, production, and shipment) of a product or a service to the customer.
- Cost of anomalies, also known as the costs of noncompliance, can be divided into two types:

TABLE 2 Definition of the CoSQ categories (adapted from Krasner 1998 and Houston 1999)

Major categories	Subcategories	Definition	Typical costs
Prevention cost	Quality basis definition	Effort to define quality, and to set quality goals, standards, and thresholds. Quality trade-off analysis.	Definition of release criteria for acceptance testing and related quality standards
	Project and process-oriented interventions	Effort to prevent poor product quality or improve process quality	Process improvement, updating of procedures and work instructions; metric collection and analysis; internal and external quality audits; training and certification of employees
Evaluation or appraisal cost	Discovery of the condition of the product nonconformance.	Discovery of the level of nonconformance	Test, walk-through, inspection, desk-check, quality assurance
	Ensuring the achievement of quality.	Quality control gating	Contract/proposal review, product quality audits, "go" or "no go" decisions to release or proceed, quality assurance of subcontractor
Cost of anomalies or nonconformance	Internal anomalies or nonconformance	Problem detected before delivery to the customer	Rework (e.g., recode, retest, rereview, redocument, etc.)
	External anomalies or nonconformance	Problem detected after delivery to the customer	Warranty support, resolution of complaints, reimbursement damage paid to customer, domino effect (e.g. other projects are delayed), reduction of sales, damage to reputation of enterprise, increased marketing

- The cost of internal anomalies: costs resulting from anomalies detected before the product or service is delivered to the customer (for example, rework).
- The cost of external anomalies: costs incurred by the company when the customer discovers defects (for example, warranty work).

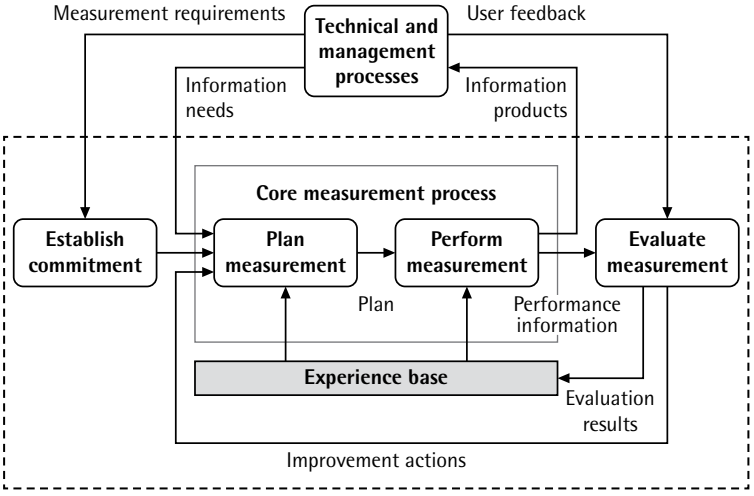
Table 2 gives the definitions of CoSQ categories, as well as typical costs for each category.

Galin also proposed to separate the share of the management's CoSQ such that management could act on improvements targeted at reducing the cost of management's activities. Galin developed the following cost items for managerial activities (Galin 2004b):

- Appraisal and control costs
  - Costs of conducting contract reviews
  - Costs for preparation of project and quality plans and their periodic updating
  - Costs for performance of regular progress control
- Failure of control costs
  - Internal managerial failure costs:
    - Unplanned development costs resulting from underestimation of resources for submitted proposals
    - Domino effect: damages to other projects planned to be performed by team members involved in delayed projects due to extra costs for recruitment of replacement team members
  - External managerial failure costs:
    - Damages paid to customers as compensation for late project completion resulting from an unrealistic schedule presented in proposal
    - Damages paid to customers as compensation for late project completion resulting from failure to recruit sufficient and appropriate team members
    - Domino effect: damages for delayed completion paid to clients of other projects planned to be performed by team members involved in delayed projects
    - Hidden external failure costs, that is, reduction of sales as a result of damaged reputation, increased investments in sales promotion, under-pricing of tender bidding to counter the effects of significant past delayed completion of projects due to managerial failures in appraisal and/or progress control tasks

As mentioned previously, a few papers have been published on the adoption of these concepts in the software industry, but very few case studies have been published by the software industry itself (Haley 1996; Galin 2007). A paper by T. J. Haley of Raytheon (Haley 1996) illustrates very well the links between the cost of rework and the investment needed to reduce waste in software projects. Haley also illustrates that a long-term perspective is needed, as well as a commitment from senior management, in order to capture the benefits. As said by Norman Augustine (Augustine 1997), "It costs a lot to build bad products."

FIGURE 3 Software measurement process of ISO/IEC 15939 (ISO 2007)



APPROACH USED TO MEASURE THE CoSQ

This section explains the steps that led to the estimated CoSQ for an 88,000-hour software development project. The authors had to develop a measurement model before proceeding with the classification of the 1100 tasks.

The ISO/IEC 15939 standard (ISO 2007) has been referred to as a reference to develop measurement models in software engineering. It describes a set of related measurement activities that are generally applicable in all aspects of a software measurement process, regardless of the specific information needs of any particular situation.

This ISO standard explains that process measurement consists of four iterative measurement activities (see Figure 3): establishing an organizational commitment, planning of measurement activities, conducting the measurement process (measurement data are collected, stored, and analyzed), and evaluating the measurement process and measures. Each activity mentioned by ISO/IEC 15939 is related to specific tasks that contribute to achieving the purpose and outcomes of the software measurement process.

The measurement model developed for the project follows various activities proposed in the ISO/IEC 15939 standard. The measurement model implementation was conducted in five stages:

- Identification of the project tasks related to the CoSQ (measurement requirements and resources)
- Development of a list of typical tasks related to the CoSQ (define data measures)
- Categorization of the tasks related to the CoSQ (select and plan measures)
- Development and application of weight factors (define measurement criteria)
- Determination of the accuracy of the weighting rules (define data collection and analysis)

TABLE 3 Representation of BSEP elements

Identification	Process element	Type
1.1	Primary life-cycle process	Life cycle
1.2	Development	Process
1.2.7	Code and debug	Subprocess
1.2.4	Unit tests	Activity

TABLE 4 Sample registration of a task

WBS element	Task name	Effort [hours]
Code document	Monitoring-unit testing	91.1

TABLE 5 CoSQ-related BSEP tasks

Primary life-cycle processes
Supply
Development
System requirements analysis
System architectural design
Software requirements analysis
Software architectural design
Software detailed design
Software coding and testing
Software integration
Software validation testing
System integration
System qualification testing
Software deployment
Supporting life-cycle processes
Configuration management
Quality assurance
Verification and validation
Peer reviews
Verification evaluation
Verification and validation reporting
Joint review
Project management reviews
Technical reviews
Problem resolution
Problem report
Organizational life-cycle processes
Management
Estimate the project
Plan the project
Manage risks
Track the project
Infrastructure
Plan infrastructure activities
Establish the project infrastructure
Establish the global infrastructure
Maintain the infrastructure
Training
Plan training activities
Implement training plan

The analysis and evaluation functions of the measurement model, and further improvements to the model, will be presented in a later section.

Stage 1: Identification of project tasks related to the cost of quality

Table 3 shows the structure of the BSEP. It is composed of the following types, as previously noted: life cycle, process, subprocess, and activity. These elements constitute the core components of the BSEP, and they are considered as inputs for the design and construction of the measurement model for the project.

Table 4 shows the organization of the work breakdown structure (WBS) for tasks related to the CoSQ for the project. Efforts expended on these tasks are identified and estimated by the project manager.

Tasks and efforts related to CoSQ for the project are identified and recorded in order to be able to measure, improve, monitor, and benchmark the CoSQ for the project.

Stage 2: Development of a list of typical tasks related to the CoSQ

At this stage, a list with the core BSEP typical tasks related to the CoSQ is developed for the project. These tasks are associated with the project life-cycle process, and are based on the ISO 12207 standard. Table 5 presents the process, subprocesses, and activities related to components of project cost and CoSQ (prevention, evaluation, and rework or nonconformance).

Stage 3: Categorization of tasks related to the CoQ

At this stage, the BSEP tasks are sorted as follows: implementation (I), evaluation (E), prevention (P), and rework (R) (internal and external anomalies). Table 6 provides a classification example for the requirements traceability task.

TABLE 6 Example of classification of the requirements traceability task

Task identification	WBS element	Task name	I	E	P	R
2410	Code	Trace requirements	-	-	x	-

TABLE 7 Task weighting rules

Rule number	Name of rule	Typical tasks	Implementation	Evaluation	Prevention	Rework
1	Normal task performance	Normal project task	100%			
2	Process improvement	BSEP, process improvement, Six Sigma			100%	
3	Process audit	Software project and process audit			100%	
4	Requirements traceability	Requirements traceability			100%	
5	Audit activities	Requirements audit		100%		
6	Prototyping	Prototype			100%	
7	Review	Design review		100%		
8	Testing activities	Software integration testing activities		100%		
9	Testing and coding	Software code and test	60%	40%		
10	Bench test	Bench test	25%	75%		
11	Validation, verification activities	Software validation, verification		100%		
12	Problem correction and coding	Corrections, debugging and final coding	30%	70%		
13	Rework	Maintenance				100%
14	Software problem correction	Software problem correction	15%			85%
15	Update SRS,SDD, DD, code, Int., validation, traceability, and SVRTM	Update SRS, traceability, and SVRTM	50%			50%
16	SQA	SQAP writing and update	30%		40%	30%
17	Training	Training of new resources			100%	
18	Configuration management activities	Configuration management			100%	
19	Analysis	Analysis	100%			
20	Test	Test		100%		
21	Bench development	Bench development	85%	15%		
22	Acceptance	Baseline acceptance		50%	50%	
23	Preparation	Release preparation	100%			
24	Working with the client	Working sessions with the client	50%		50%	
25	Follow-up	Follow-up (all releases)	75%		25%	
26	Follow-up and validation	Follow-up and validation (all releases)	85%			15%
27	Acceptance, debug and test prototype	Acceptance, debug and test prototype	100%			

Stage 4: Development of weighting rules

In the BSEP, there are activities that belong to more than one category; for example, the test and coding task overlaps the evaluation and implementation CoSQ categories. To ensure that one measures correctly, weighting rules have been defined. Twenty-seven rules, as illustrated in Table 7, have been defined in close collaboration with SDG engineers. As an example, rule 9 for the test and coding task has been assigned a weight of 60 percent for implementation and 40 percent for evaluation.

TABLE 8 Examples of cost of software quality data for three tasks

Task name	Effort measured (hours)	I	E	P	R
Software problem correction	883	132	0	0	751
Train simulator-software code and test	195	117	78	0	0
Baseline acceptance	24	0	12	12	0

TABLE 9 Level of confidence

Rule number	Rule name	Number of occurrences
1	Performing normal task	492
2	Process improvement	6
3	Process audit	10
4	Requirements traceability	9
5	Audit activities	47
6	Prototyping	22
7	Review	41
8	Testing activities	36
9	Testing and coding	145
10	Bench test	5
11	Validation, verification activities	93
12	Problem correction and coding	8
13	Rework	11
14	Software problem correction	36
15	Update SRS, SDD, DD , code, Int., validation, traceability, and SVRTM	57
16	SQA	3
17	Training	5
18	Configuration management activities	3
19	Analyses	3
20	Test	3
21	Bench development	2
22	Acceptance	25
23	Preparation	27
24	Working with the client	12
25	Follow-up	2
26	Follow-up and validation	3
27	Acceptance, debug and test prototype	1
	Total	**1107**

Stage 5: Determining the accuracy of weighting rules

The wide variety of actions involved in conducting any activity led to the following question: At what level of confidence should one measure CoSQ? To answer this question, the authors added another component to the precision of the rules weighting in the model of quality in the form of a convention, adopted at task registration:

- "H" for high precision
- "M" for medium precision
- "L" for low precision

The rules are now weighted, and the result of this process is expressed as a percentage of tasks for each level of confidence.

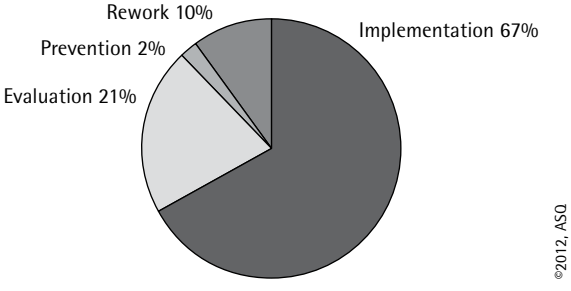
Data Model for the Measurement of CoSQ

The data model is a compilation of all of the components previously presented, namely the identification of activity/confidence level.

This data model is set out in a spreadsheet that includes, among others, the following four tabs:

- Data – Task: This tab includes the data that characterize: task name, effort, status, number of the rule, precision, display of the rule, and measure of the effort of the task (see Table 8).
- Analysis – Task: This tab presents the analysis using the spreadsheet. Histograms give a first performance of the extended CoSQ.
- BSEP versus CoSQ: This tab shows the CoSQ activities compared to the BSEP process.

FIGURE 4 Distribution of effort in the 88,000-hour project



- Tasks types (task): This tab shows the types extracted from the task table and the link with the BSEP. These activities are associated with CoSQ categories.

PRESENTATION AND DISCUSSION OF RESULTS

The Cost of Software Quality

As illustrated in Table 9, the level of confidence assigned to each rule has resulted in more than 88 percent of the activities being in the high category, 11 percent in the medium category, and only 0.2 percent in the low category. One can therefore conclude that the CoSQ was measured with a good level of accuracy.

Cost of Software Quality Per Category

Figure 4 shows the distribution of development costs in the various categories of software quality and implementation cost. The figure reveals that the cost of rework is 10 percent, the cost of prevention is 2 percent, and the cost of evaluation is 21 percent of the total cost of development.

At Raytheon (Haley 1996), Haley illustrated the relationships between the investments/benefits and the maturity levels of the Capability Maturity Model for Software (CMM®) (Paulk et al. 1993). At the initial CMM maturity level, most, if not all, organizations have no documented process, and no reliable data or measurement system. When an organization reaches levels 2 and 3, it has the foundation to measure and select beneficial process improvement areas. Many organizations have published the results obtained in the form of a maturity ladder, showing the relationships between maturity level, quality, productivity, and project cost. The study at Raytheon showed (see Figure 5) that, at CMM level 1, the cost of rework in 1987 was about 41 percent of total project cost. Rework was 18 percent at level 2, 11 percent at level 3, and 6 percent at level 4.

FIGURE 5 CoSQ data at Raytheon (adapted from Haley 1996)

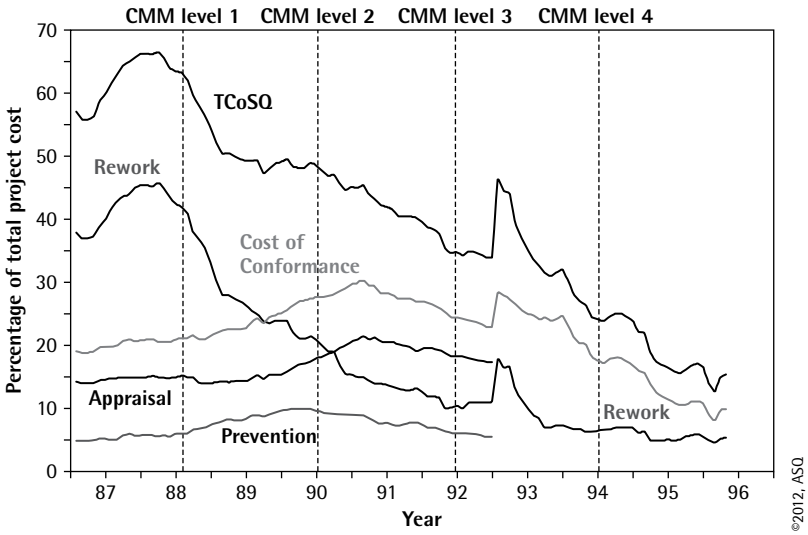


TABLE 10 Relationship between CMM maturity levels and cost of rework (Gibson et al. 2006)

CMM maturity level	Percentage of rework
1	≥ 50%
2	25% to 50%
3	15% to 25%
4	5% to 15%
5	≤ 5%

FIGURE 6 CoSQ at the Motorola Global Software Group Center in China (Liu 2007)

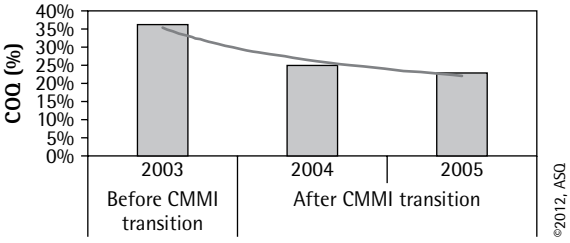


TABLE 11 Cost of software quality data from software professionals and managers

	Site A American Engineers (19)	Site A American Managers (5)	Site B European Engineers (13)	Site C European Engineers (14)	Site D European Engineers (9)	Course A 2008 (8)	Course B 2008 (14)	Course C 2009 (11)	Course D 2010 (8)	Course E 2011 (15)	Course F 2012 (10)
Cost of performance	41%	44%	34%	31%	34%	29%	43%	45%	45%	34%	40%
Cost of rework	30%	26%	23%	41%	34%	28%	29%	30%	25%	32%	31%
Cost of appraisal	18%	14%	32%	21%	26%	24%	18%	14%	20%	27%	20%
Cost of prevention	11%	16%	11%	8%	7%	14%	10%	11%	10%	8%	9%
Quality (Defects/ KLOC)	71	8	23	35	17	403	19	48	35	60	55

A study by Gibson, Goldenson, and Kost (2006) showed that rework varies between 15 percent and 25 percent of the cost of developing CMM maturity level 3 (see Table 10). The study (Gibson, Goldenson, and Kost 2006) shows how the implementation of CMMI impacts schedule, cost performance, product quality, return on investment, and other factors in organizations. In the case of the Motorola Global Software Group in China (Liu 2007), for example, it was possible to reduce the CoSQ by more than one third (36.5 percent) from its pre-CMMI baseline (see Figure 6).

One of the authors of this article collected data on the CoSQ in their environment from professional engineers, managers, and software professionals, working in a wide range of application domains, enrolled in the software engineering master's program at the ETS engineering school of Montréal (www-eng.etsmtl.ca). As illustrated in Table 11, the cost of rework is about 30 percent. Most of the industrial data were collected in two large multinational enterprises: one involved in the transportation sector and the other in the aerospace sector. The numbers in parentheses indicate the number of people responding to the CoSQ survey form.

With regard to the evaluation and prevention categories, a study by Price Waterhouse (Price Waterhouse 1988) showed the effort required for quality control: the sum of the evaluation cost and the prevention cost is between 23 percent and 34 percent of the total development cost. Based on this result, for the SDG group, the cost of quality control is 23 percent of the total cost of development, which is consistent with previous studies.

TABLE 12 Distribution of project effort

	Implementation	Evaluation	Prevention	Rework
Number of hours per category	56,282	20,070	3,142	9,103
Percentage of project	68%	17%	4%	11%

Discussion of Results

Table 12 shows that the CoSQ represents 33 percent of the overall project effort of more than 88,000 hours. The study conducted by Price Waterhouse shows that the CoSQ varies between 38 percent and 49 percent of the total cost of development.

This study, however, excludes the cost of testing and the resulting anomalies, thereby reducing the CoSQ, estimated to be between 40 percent and 55 percent of the project cost. The study, conducted at Raytheon, shows that the CoSQ fluctuated between 55 percent and 67 percent when the company was at maturity level 1, while it decreased to 40 percent when it reached maturity level 3. These data from the software industry validate the results obtained

by the SDG. It should be noted that the CoSQ of the SDG is lower than that estimated in the two previous studies. This is because its processes were assessed to be at SEI's CMM level 3. Several software components developed by this group are critical; that is, if faulty, they can cause fatal injuries. The development of critical software requires the addition of several prevention activities, such as testing.

FIGURE 7 Distribution of effort by weighting rules

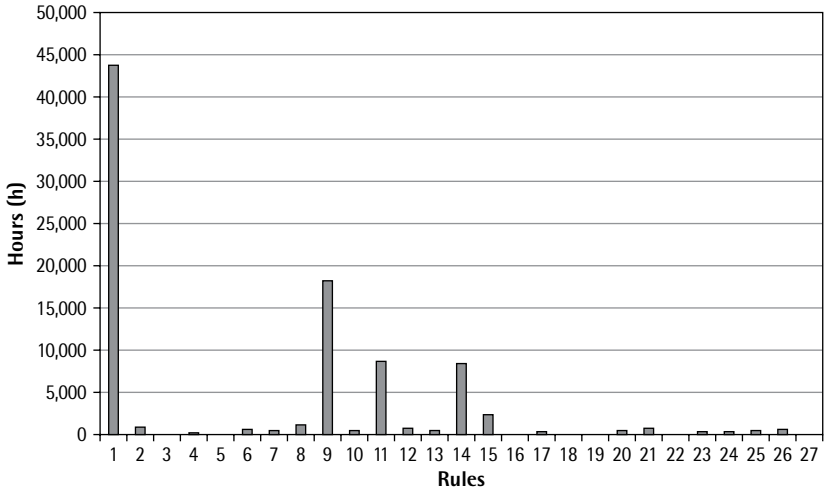
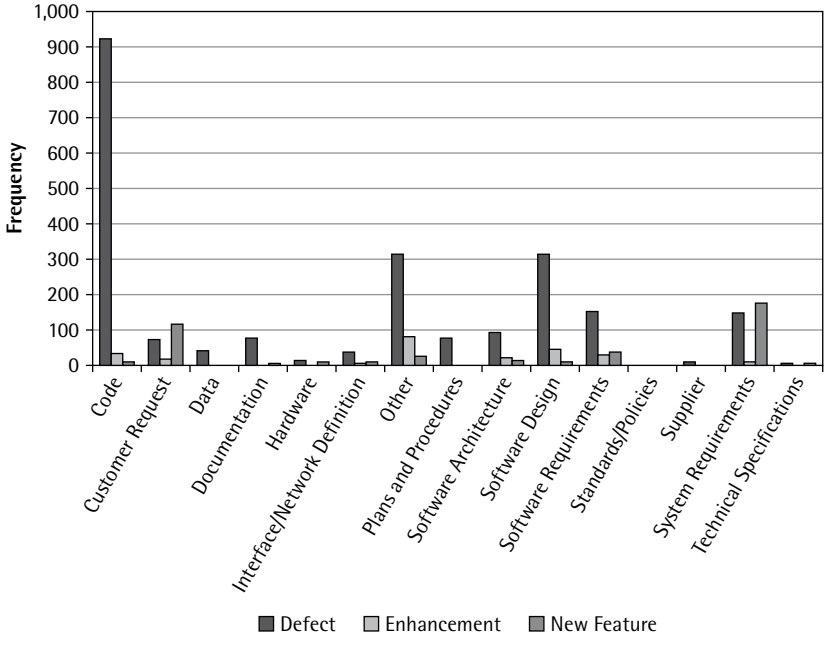


FIGURE 8 Distribution of effort by type of defect and associated activity



Calculation of the Compliance/ Noncompliance Ratio

The ratio of compliance to noncompliance gives the CoSQ and gives the ratio between the CoSQ and the cost of the rework. The Price Waterhouse study presents a ratio of 1.2 to 2.0. For this project, the authors obtained a ratio of 2.1.

The difference between these two studies may be explained by the fact that the SDG's cost assessment is quite high. The authors recommend lowering the cost of assessment, while at the same time making improvements in the area of prevention. The high rate of prevention can be explained by the fact that the software developed by Bombardier Transportation is sometimes critical and demands more prevention activities.

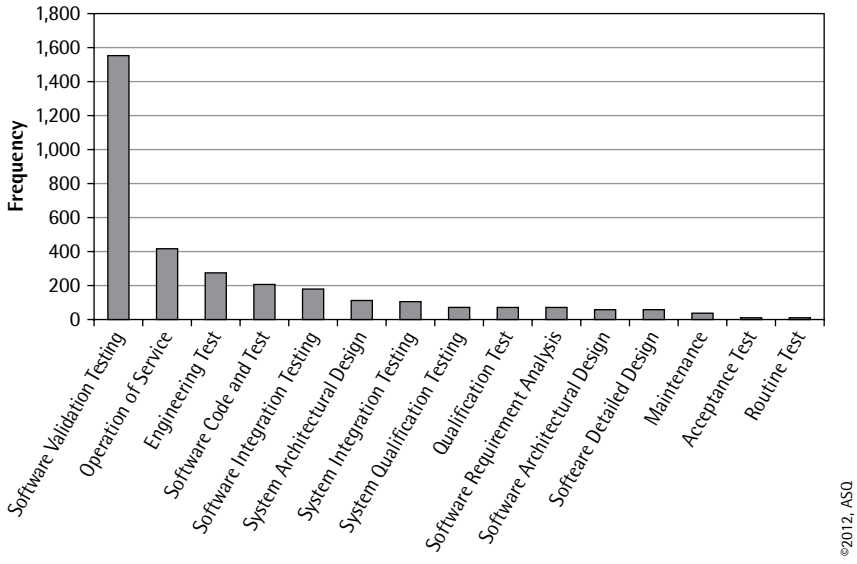
Distribution Costs of Software Quality by the Rules of Weighting

To measure the CoSQ, the authors applied the rules of weighting to each task of a software project. The analysis of the distribution of cost versus weighting

allows the examination of the activities that have the greatest impact on the CQL.

Figure 7 shows that the most significant costs are associated with rules 8, 9, 11, and 13 and, to a lesser extent, rules 2, 6, 8, 12, 13, 15, 20, and 21, while rule 1 represents the implementation cost. This is the explanation of why the cost of evaluation is high compared to other categories. Rules 8 and 9 are associated with testing activities and coding; while rule 8 gives 100 percent of the category evaluation, rule 9 provides 60 percent of the cost in terms of achievement category and 40 percent at the category assessment. Rule 11 shows the activities of validation and verification (that is, activities of assessments) that are represented at 100 percent in the category evaluation.

FIGURE 9 Distribution of the frequency of activities, depending on the type



Rules 13 and 14, associated with recovery activities and correcting software problems, essentially form the category of rework. Rules 2 and 6, respectively, representing a business process improvement activity and a prototype, belong to the category prevention. Rule 25, which is a follow-up, is weighted at 75 percent implementation and 25 percent for prevention.

This analysis shows that, if one wants to affect the cost associated with each category, either increasing it or decreasing it, one must do so through actions on the activities mentioned previously (for all rules).

Cost of Rework Relative to the Cost of Correcting Problems

In order to analyze the categories and activities related to the CoSQ, the authors studied the relationship between the type of activity and the cause of a problem, as recorded on the problem report form filled out by the SDG engineers. The three causes of a problem are the following: a defect, an improvement or an enhancement, or a new feature. Figure 8 shows the distribution of costs based on the nature of the problem and its origin (for example, code, design).

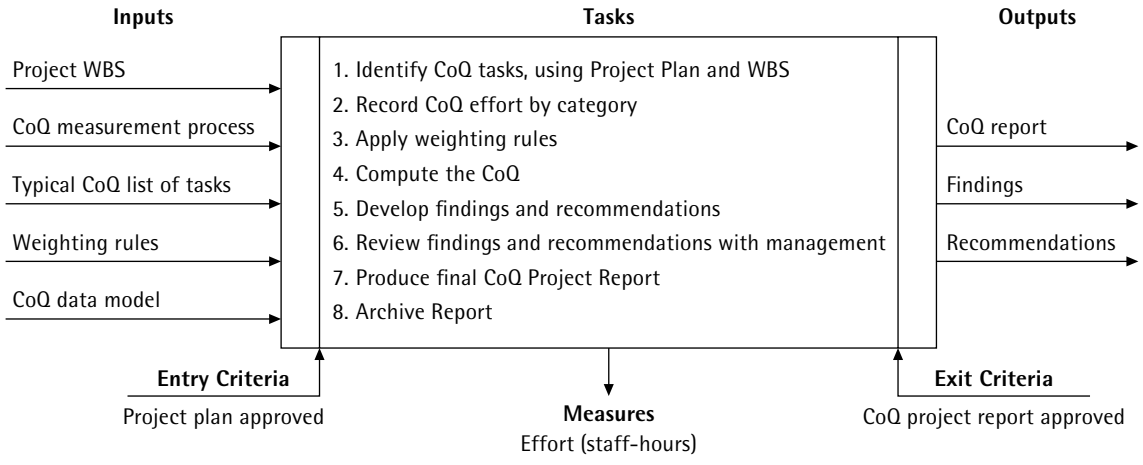
This cost allocation shows that the problems occurring during the development cycle affect three components: code, design, and software specifications, while the main source of the problems lies in the coding phase. Although they are not high enough, the problems associated with a "new feature" must be addressed, and are more commonly found at the level of customer requests and requirements, and software system requirements. For issues like improvements, their contributions to the cost of correction remain low, but it is important that they continue to be monitored.

Thus, to better control these costs, the activities that will be most involved in calculating the cost for correcting problems must be identified, and their sources addressed.

Moreover, an important factor to consider in decision making is the frequency with which activities are performed, and the effect that increased investment would have, which gives an idea about the overall cost of activity, for example, evaluation activities. This could be an avenue for future research. Figure 9 shows the distribution of the frequency of activities.

This distribution, which provides a measure of the attention paid to activities, can reveal where discrepancies lie. Of all the activities of the development process, the software validation and testing activity should

FIGURE 10 Top level modified ETVX description of the process to measure the CoSQ



command more attention on the part of the SDG, with more than 1500 hours spent on testing. Is there a way, for example, to add prevention activities while reducing testing and at the same time maintaining at least the same level of quality?

RECOMMENDATIONS

The authors' recommendations, following an analysis of the data on the CoSQ, are presented here. These can help control spending on the CoSQ, as well as the cost of development or maintenance of software.

1. **Continue data collection.** The first recommendation is to continue data collection using procedures and methods (for example, use of tools, database structures) within the SDG.
2. **Continue CoSQ measurement.** The rate of 33 percent, representing the cost incurred by the software quality group SDG, is comparable to rates commonly found in the software industry. Studies conducted by Price Waterhouse, as well as by Dion (1992) and Krasner (1998), both confirm and validate the data model that is introduced here. However, it is important to note that continuing to measure CoSQ would be a good initiative. It is important to control the overall CoSQ, but without making budget cuts to CoSQ-related activities. The best would be distributing the CoSQ over the categories.
3. **Control CoSQ by category.** With regard to categories like the cost of prevention, the rates should be better controlled. However, the rate of quality control (that is, the rate of assessment and the cost of prevention combined), which is 23 percent, is validated by the studies cited previously.

   Since the CoSQ ratio in terms of conformity to quality is outside the 1.2 to 2.0 range, the authors recommend that the rate of quality control of 20 percent be reduced. This could be achieved by increasing the budget for prevention activities. This will result in an increase in prevention activities, a reduced rate of assessment, and, therefore, a better quality of products delivered.

   The analysis gives the highest rates of rework at 10 percent of the total cost of development. The authors recommend that this rate be maintained and that the effort to change the rate of assessment and prevention be strengthened, which will lead to lower recovery efforts.

   They also recommend that the names given to tasks and activities be harmonized and standardized, so when they are registered, the errors that have sometimes been made in applying the rules to the CoSQ can be avoided.
4. **Control activities related to the correction of problems.** Since defects are found mainly at the level of software design, software

requirements, and code, it would be wise to encourage actions to reduce these by focusing on prevention, a view that supports the recommendation made previously for increased investment in this category. Regarding the two categories of rework, internal and external, it would be beneficial to undertake appropriate remedial actions such as improving the efficiency of anomaly detection by implementing peer reviews.

5. **Present CoSQ measurement results.** The authors recommend that the results of the work undertaken in this study be presented to the entire project team with the aim of providing it with avenues for future research. It would be desirable to establish a scoreboard of the costs associated with software quality and an improvement program showing the link between the activities of the program and the CoSQ. These results should be presented to management to enable them to develop better budgets for business process improvement.
6. **Measure CoSQ at other Bombardier Transportation sites.** The authors recommend that an individual in the software quality assurance group or the process improvement group be permitted to launch a measurement CoSQ. The objectives of the implementation of this process would be:
    - To develop software according to a CoSQ measurement process
    - To quantify the components of the CoSQ
    - To identify opportunities for reducing costs
    - To provide a basis for budget development activities and quality
    - To use the results to improve processes

The main tasks, inputs, and outputs of the CoSQ measurement process are described in Figure 10 using the modified ETVX "Entry, Task Validation eXit" process notation (Radice 1985).

FUTURE WORK

The measure of CoSQ presented in this article is based on data from a single large project. In order to improve the measurement of CoSQ and achieve management goals more effectively (as discussed at the beginning of this article), it would be appropriate to:

- Check the definitions of the words composing the names of elementary tasks used in the capture of effort during the project. This will have a significant impact in terms of reducing the number of weighting rules, which will in turn allow better control of the activities and of the CoSQ.
- Redefine and improve the accuracy of some of the weighting rules.
- Assess the level of satisfaction of high-level management by introducing the CoSQ to the SDG and analyze its impact in terms of improving other processes.
- Measure the CoSQ for other SDG projects and compare the results.
- Find the extent of CoSQ implementation at other Bombardier Transportation sites. Collect data from those sites and conduct a comparative analysis based on several criteria, such as the type of software, organizational culture, the size of the development group, the level of maturity, and so on.

A very small entity (VSE) is defined in (ISO 2011a) as an enterprise, organization, department, or project having up to 25 people. VSEs developing software are very important to the world economy, as their software components are often integrated into the products of larger entities. Many international standards and models have been developed to capture proven engineering practices. However, these standards and models were not designed for VSEs and are consequently difficult to apply in such settings. An ISO Working Group, ISO/IEC JTC1 SC7 WG24, was established in 2005 to help VSEs by developing a set of software engineering standards and guidelines that are specifically tailored to the needs of VSEs (Laporte, Alexander, and O'Connor 2008a; 2008b). A standard for VSEs, ISO/IEC 29110 (ISO 2011b), and a free engineering and management guide (ISO 2011c) have been published recently. Most of the CoSQ data illustrated in Table 11 were collected from software professionals working in VSEs. The work described in this article could be used by the ISO working group to help VSEs measure the CoSQ and guide them in reducing the amount of rework.

CONCLUSIONS

In the field of software engineering, the concept of the measure has become very important, and is, in fact, fundamental, as it is considered a measure of a company's performance. The ISO/IEC 15939 standard is a valuable reference for implementing the software measurement process in companies. This study presented the Bombardier Transportation SDG's measure of the CoSQ.

The authors added weighting rules to their measurements, which identify the level of confidence of these measurements by quantifying their accuracy. They intend to improve these rules in future work. Analysis shows that the CoSQ represents 33 percent of the overall project cost. To make an objective comparison, this analysis looked at steps taken by researchers in companies at the same level of maturity. This comparison shows that the CoSQ of the project is similar to published data, or even slightly lower than these data. This can be explained by the fact that the SDG process conformed to level 3 maturity for more than four years. The other aspect that cannot be overlooked is the cost of takeovers, which identify activities that have the greatest impact in terms of correcting problems.

It would be equally beneficial to redefine and improve the accuracy of some of the weighting rules, and it would be interesting to control the definition of tasks in the capture of effort data during a project. This would have a significant impact in terms of reducing the number of weighting rules, which would in turn allow better control of activities and CoSQ. It also remains important to assess the level of satisfaction of high-level management resulting from the introduction of CoSQ SDG and analyze its impact in terms of improving the process.

Other avenues for improving the CoSQ process are possible, such as staff training, automation testing, and so on. Other Bombardier Transportation sites, as well as many VSEs, could benefit from the experience gained from the CoSQ results of this project.

ACKNOWLEDGMENTS

The authors acknowledge the invaluable collaboration of Yves Laperrière and Sylvain Hamel of Bombardier Transportation in St-Bruno (Québec).

REFERENCES

Augustine, N. 1997. *Augustine's laws,* sixth edition. Reston, VA: AIAA.

Campanella, J. 1990. *Principles of quality costs,* 3rd ed. Milwaukee: American Society for Quality Control.

Charette, R. N. 2005. Why software fails. *IEEE Spectrum* 42, issue 9.

Dion, R. 1992. Elements of a process improvement program, Raytheon. *IEEE Software* 9, no. 4 (July):83-85.

Dion, R. 1994. *Elements of a Successful Process Impovement Paradigm.* Presentation to the Montreal Software Process Improvement Network (SPIN), Montréal, Canada, 25 May.

Feigenbaum, A. V. 1956. Total quality control. *Harvard Business Review* 34, no. 6 (November-December):93-101.

Galin, D. 2004a. *Software quality assurance—From theory to implementation.* Upper Saddle River, NJ: Pearson Education Limited.

Galin, D. 2004b. Toward an inclusive model for the costs of software quality. *Software Quality Professional* 6, no. 4:25-31.

Galin, D., and M. Avrahami. 2007. Benefits of a higher quality level of the software process: Two organizations compared. *Software Quality Professional* 9, no. 4 (September):27-35.

Gibson, D. L., D. R. Goldenson, and K. Kost. 2006. P*erformance results of CMMI®-based process improvement.* Technical Report (CMU/SEI-2006-TR-004, ESC-TR-2006-004). Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

Haley, T. J. 1996. Software process improvement at Raytheon. *IEEE Software* 13, no. 6:33-41.

Houston, D. 1999. Cost of software quality: Justifying software process improvement to managers. *Software Quality Professional* 1, no. 2 (March):8-16.

ISO/IEC 15939:2007. 2007. *Information Technology–Software Engineering–Software Measurement Process.* Geneva, Switzerland: International Organization for Standardization.

ISO/IEC 29110-1:2011. 2011a. Software Engineering—Lifecycle Profiles for Very Small Entities (VSEs)—Part 1: Overview. Geneva: International Organization for Standardization (ISO). Available at: http://standards.iso.org/ittf/PubliclyAvailableStandards/c051150_ISO_IEC_TR_29110-1_2011.zip.

ISO/IEC 29110-4-1:2011. 2011b. Software Engineering—Lifecycle Profiles for Very Small Entities (VSEs)—Part 4-1: Profile specifications: Generic profile group. Geneva: International Organization for Standardization (ISO). Available at: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51154.

ISO/IEC TR 29110-5-1-2:2011. 2011c. Software Engineering—Lifecycle Profiles for Very Small Entities (VSEs)—Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile. Geneva: International Organization for Standardization (ISO), 2011. Available at: http://standards.iso.org/ittf/PubliclyAvailableStandards/c051153_ISO_IEC_TR_29110-5-1_2011.zip.

Karg, L. M., M. Grottke, and A. Beckhaus. 2011. A systematic literature review of software quality cost research. *Journal of Systems and Software* 84, no. 3 (March):415-427.

Krasner, H. 1998. Using the cost of quality approach for software. *Crosstalk—The Journal of Defence Software Engineering* 11, no. 11 (November).

Laporte, C. Y., S. Alexandre, and A. Renault. 2008a. The application of international software engineering standards in very small enterprises. *Software Quality Professional* 10, no. 3, (June).

Laporte, C. Y., S. Alexandre, and R. O'Connor. 2008b. A software engineering lifecycle standard for very small enterprises. EuroSPI 2008, CCIS 16, 129-141,

Liu, Angel Qi. 2007. Motorola Software Group's China Center: Value added by CMMI, Data & Analysis Center for Software, Department of Defense. *Software Tech News* 10, no. 1 (March):19-23.

Mandeville, W. A. 1990. Software cost of quality. In *Proceedings of the IEEE Journal on Selected Areas on Communications* 8, no. 2:315.318.

Paulk, M., B. Curtis, M. B. Chrissis, and C. Weber. 1993. Capability Maturity Model for Software, version 1.1 (CMU/SEI-93-TR-24). Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

PMI. 2008. *A Guide to the Project Management Body of Knowledge* (PMBOK® Guide), fourth edition. Newton Square, PA: Project Management Institute.

Price Waterhouse Report. 1988. *Software quality standards: The costs and benefits.* UK Department of Trade and Industry.

Radice, R. 1985. A programming process architecture. *IBM Systems Journal* 24, no. 25.

Slaughter, S.A., D. E. Harter, and M. A. Krishnan. 1998. Evaluating the cost of software quality. *Communications of the ACM* 41, no. 8:10.17.

BIOGRAPHIES

Claude Y. Laporte is a professor, since 2000, at the École de technologie supérieure, an engineering school, where he teaches software engineering. His research interests include software process improvement in small and very small organizations and software quality assurance. He has worked in defense and transportation organizations for more than 20 years. He is the editor of the ISO/IEC JTC1 SC7 Working Group 24 tasked with developing ISO/IEC 29110 systems and software engineering standards and guides for very small entities. He is a member of INCOSE, IEEE, PMI, and the professional association of engineers of the Province of Québec. He is the co-author of two books on software quality assurance. Laporte can be reached by email at Claude.Y.Laporte@etsmtl.ca.

Nabil Berrhouma has more than 15 years of experience in software development. He has experience in many aspects of IT and the software development life cycle. He is a business and functional analyst. Berrhouma has helped organizations develop and implement SDLC methodologies, especially in the analysis and quality fields. He has a master's degree in software engineering from the École de technologie supérieure of Montréal (Canada). He is interested in development and operational processes, and he is a co-author of an article about ITIL.

Mikel Doucet is an international director for Bombardier Transportation. He is responsible for the ORBIFLO™ wayside management product development and commercialization. Previously he was responsible for the Center of Competence in Software Engineering at Bombardier Transportation. This Competency Center covers nearly 1000 people in 12 countries and was responsible for defining and deploying common processes and tools, performing risk assessment, addressing strategic mandates, and supporting bids/projects. Before joining Bombardier, Doucet held various software engineering positions in industry. He received a master's degree in engineering management and a bachelor's degree in mechanical engineering from the University of Sherbrooke (Canada).

Edgardo Palza-Vargas has more than 15 years of experience as a professor, researcher, and consultant. His concentration covers domains such as software engineering, MIS, and TI. Palza's current research activities cover artificial intelligence applied to management (health, finance, and marketing), business intelligence (data mining/data warehouse), IT governance, and business strategies. He has published several books, chapters, and articles in journals and international conferences. He is a member of the editorial board, a reviewer, and chairperson for international conferences. Palza has implemented business and IT solutions for organizations such as Nokia, Ericsson Research Canada, John Hancock, NASA IV&V, World Bank, Bombardier, Cirque du Soleil, and CGI.