Journal of Information Technology Management

ISSN #1042-1319

A Publication of the Association of Management

IMPLEMENTING QUALITY GATES THROUGHOUT THE ENTERPRISE IT PRODUCTION PROCESS

VLADIMIR AMBARTSOUMIAN
UNIVERSITY OF MEMPHIS
vgmbrtsm@memphis.edu

JASBIR DHALIWAL
UNIVERSITY OF MEMPHIS
jdhaliwl@memphis.edu

EUNTAE "TED" LEE
UNIVERSITY OF MEMPHIS
elee@memphis.edu

THOMAS MESERVY
UNIVERSITY OF MEMPHIS
tmeservy @memphis.edu

CHEN ZHANG
UNIVERSITY OF MEMPHIS
czhang12@memphis.edu

ABSTRACT

The concept of quality gates has been successfully applied as a quality assurance mechanism in several industries. The quality gates approach combines aspects related to project management, decision modeling, and work flow management to increase measurability and promote quality. Software development and testing organizations are now investigating the use of this approach for the purposes of promoting software quality and improving software development processes. This paper summarizes prior literature on quality gates and applies the concept of quality gates to the software development context. It reports a case study of implementing quality gates in enterprise IT production process in the context of a large Fortune 500 company. A conceptual framework is also proposed to represent the various levels and disciplines where quality gates may be implemented. This framework suggests that 1) quality gates can be applied at many different levels throughout the organization such as system, project, and release, 2) the format of quality gates varies by level of implementation, 3) quality gates are useful both as part of an overarching software development methodology as well as for targeted IT projects where quality assurance standards have to be established for procedural success (e.g.,, transitioning enterprise data centers), and 4) more than the increased measurability promoted by quality gates, greater value may be yielded by a change in developers' and testers' mindset of building quality into the software product and development processes.

Keywords: quality gates, software development, software quality, quality assurance, software development life cycle, enterprise IT production process

INTRODUCTION

Most organizations are dependent on highly complex software packages to achieve their organizational/business objectives. Although the salient question for some of these organizations is which software to buy and how to configure and integrate it with existing systems, many organizations (including many Fortune 500 companies that use technology to sustain their competitive advantage) still must develop custom applications that need to work in concert with numerous other systems that have been developed in-house and/or acquired from software vendors. In such an environment, assuring software quality is a non-trivial task, and thus over the years numerous techniques have been introduced in an attempt to increase software quality. One such technique is the Quality Gates approach, which has been successfully applied in many industries for quality assurance during production processes.

This paper reviews literature on the origin of Quality Gates, the use of Quality Gates in manufacturing processes, and benefits and challenges in using Quality Gates. What follows is a discussion of how to combine the stage-gate project management model with various software development methodologies and how to use Quality Gates in the different phases of the systems development lifecycle. Then, a case study of implementing quality gates in enterprise IT production process is presented and a conceptual framework is proposed. The paper concludes with a discussion of the contributions and implications of our study.

OVERVIEW OF QUALITY GATES

Numerous quality assurance approaches have been adopted to increase the quality of the ultimate output of a production process. A relatively common technique is the Quality Gates approach. Quality Gates, when used in conjunction with project management, help manage the balance between cost, functionality, and quality.

Literature regarding Quality Gates as a Quality Management concept originated in the European Union. Quality Gates were initially applied to product development processes especially quality control in the automotive industry. Since then, Quality Gates have been more broadly applied to quality assurance and project management [20] [23].

The concept of Quality Gates is based on the stage-gate system initially presented in 1986 and later refined by other researchers (e.g., [22]). In order to apply this technique, a process must be broken down into several distinct phases. Then, quality checkpoints or gates are placed between phases as an explicit means of checking the quality of artifact that is being produced. Figure 1 [5] illustrates a simple example of a production process that incorporates stage gates.

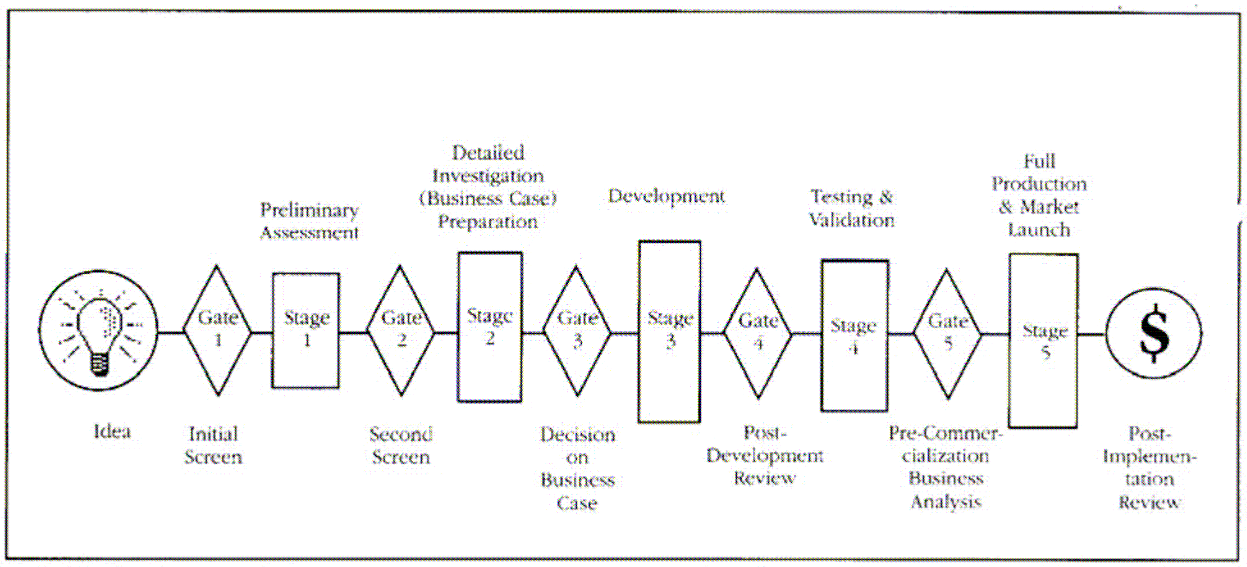

Figure 1: Overview of a Stage-Gate System

In general, a Quality Gate marks the formal end to a particular process within a project, a "gate" through which the project proceeds from one phase to another. Each gate results in the certification that all appropriate work required to move products forward to subsequent project activities has been completed and reviewed and products meet specific quality expectations. Based on a set of pre-determined exit criteria established for each phase or milestone being certified, a Quality Gate results in a pass/fail decision for moving forward [6].

Over the years many different conceptualizations of Quality Gates have been proposed. Table 1 captures some of these perspectives. While there is no universally accepted definition across industries as to what a Quality Gate is or how it should be structured, certain regional or industry initiatives have emerged including the Transregional Collaborative Research Centre SFB/TR4 described in academic literature [21] and the trade press [3].

Table 1: Quality Gates Definitions

Author	Definition
Charvat (2003)	• Formal checklists are used throughout the life of a project. • Formal sign-off and acceptance occurs at each gate. • Assessment of the quality and integrity of the product takes place. • Information is assured to be communicated to the correct stakeholders (i.e., deployment hands off to operations, etc.).
Flohr (2008)	• Significant milestones and decision points with predefined and quality focused criteria
Schneider (2004)	• "Quality Gate is a checkpoint consisting of a set of predesigned quality criteria that a project must meet in order to proceed from one stage of its life cycle to the next. Quality Gates thus serve as amendments to milestones and deliverables which meet predefined quality benchmarks"

Quality Gates require the definition of specific entry and exit criteria. One typical criterion is the completion of deliverables to be exchanged between different phases of a process. Although Quality Gates are similar to milestones in that certain results are expected, Quality Gates are more flexible as they are not tied to a particular timeframe. Quality Gates can also serve as a point of synchronization of process results and entry and exit criteria must be met before the product is able to continue throughout the process. Quality Gates help to break down the overall requirements on the final process result into sub-targets for the single process steps and to clarify the internal dependencies of the process chain [25].

Historically Quality Gates have been viewed as binary checkpoints throughout a serial process where all of criteria have to pass in order to continue on to the next activity. However, Quality Gates criteria can also include the success of other Quality Gates in such a way that Quality Gates can be interconnected with each other [23]. Additionally, Quality Gates need not run only serially, but can (and often) run in parallel. That is, different sub-processes run independently but at some point filter together as products outputted from one phase are used as inputs for the next phase. Additionally, it is important to recognize that Quality Gates are often implemented on a very granular scale that they can be implemented at different levels of process abstraction. For example, a code-level Quality Gate might be established for assuring the quality of a piece of software code that a developer is implementing and at the same time a higher-level project-level Quality Gate may be established for determining whether or not the software project should be released on a particular date.

Numerous benefits of adopting Quality Gates have been identified in the extant literature. These benefits can generally be dichotomously categorized as benefits associated with the product itself or benefits to the production process. For instance, a benefit associated with the product would be the enhanced quality of the product as a result of a series of Quality Gates that were implemented. The production process also sees direct benefits from Quality Gates such as increased communication between teams. Table 2 summarizes some of the benefits of implementing Quality Gates as identified in the literature.

Table 2: Quality Gates Benefits

Category	Benefit	Description	Source
Product	Ability to assess quality	Q-gates provide a mechanism for the project team to readily assess the quality of their work products throughout the project life cycle and improve quality at the source. Improve status visibility Measure the current project status more efficiently and effectively Q-gates ensure that intermediate work products meet the needs of downstream activities through a formal, disciplined process. Problems, their resolution, and opportunities for improvement are identified; and risk assessment or escalation procedures are invoked, if appropriate. The quality of the single process step output is measurable and controllable for every process participant,	Younack [27] Schneider [23] Schneider [23] Younack [27] Valeri and Rozenfeld [24]
Product	Decreased project risk	Minimizing project risk through phase-by-phase checklists	Charvat [3]
Process	Reduced development time	Reducing development cycle time—getting it done right the first time	Charvat [3]
Process	Enhanced project communication	Q-gates enhance project communications, thereby resulting in improved management of the expectations of key stakeholders through their participation in Q-gate certifications. Enabling project managers to continuously communicate the process and build quality directly into the project	Younack [27] Charvat [3]
Process	Focus on Quality	Increasing focus by project team on a well-designed product	Charvat [3]
Process	Better planning and control	Better support planning The Quality Gate concept helps to overcome the complexity in the planning and control of production process chains Better control necessary changes or improvements	Schneider [23] Valeri and Rozenfeld [24] Schneider [23]
Process	Shared Responsibility	Project stakeholders share responsibility with the project manager for the project's quality outcome.	Younack [27]

Despite the salient benefits of using Quality Gates, there are still numerous challenges associated with implementing Quality Gates. For example, in order to be able to implement Quality Gates, production processes must be decomposed into discrete phases. In addition, each Quality Gate is often characterized by its entry and exit criteria. Because in many environments the production process spans teams, departments, or organizations, agreeing on useful entry and exit criteria can be challenging as all parties may not share a common vision of what is requisite for quality at a particular stage or may be biased due to differing perspectives/cultures. More importantly, a major task for organizations that implement Quality Gates is determining where during the production process they should be implemented, how to structure and define Quality Gate criteria and how Quality Gates relate to each other. We suggest that organizations address at least the following questions prior to implementing Quality Gates:

- Where should Quality Gates be implemented?
- What types of Quality Gates should be implemented?
- What should the entry and exit criteria be for each Quality Gate?
- Who is responsible for monitoring each Quality Gate?
- How will the results of the Quality Gate criteria be used in decision making related to the subsequent stages of the production process?

QUALITY GATES IN SOFTWARE DEVELOPMENT

A few researchers have outlined combining stage-gate project management models with various software development methodologies. For example, Karlstrom and Runeson [12] investigate the introduction of an agile software development process within a traditional stage-gate model and identified challenges as well as success factors in the integration process. Wallin et al. [26] also propose an integration of software development life-cycle models (SDLM) and business decision models such as Cooper's Stage-Gate Process Model by demonstrating that the latter can be mapped to the major milestones in both the unified process and extreme programming. Finally, Nguyen [13] proposes a decision model for managing software development projects. The motivation of this study is somewhat similar to the study of Wallin et al [26] in that both studies try to bridge the gap between SDLMs and business decision-making. The decision model consists of a set of indicators, which link tasks performed during each phase of the software development process (i.e., project definition, requirements, design, and implementation) and the responsible managers. The quality of the tasks is assessed based on their completeness, correctness, consistency, and compliance. The model is then illustrated using a prototypical web-based implementation. The above studies generally discuss the integration of stage-gate models and software development life-cycle models at a high level with limited details on how Quality Gates are actually implemented (e.g., types of gates, gate criteria).

From a practical standpoint, a number of researchers and practitioners have provided some guidelines for implementing Quality Gates in each phase of the SDLC. For example, Tarrani and Zarate [24] break the software development process into nine distinct phases: requirements, design, build, product test, quality assurance, roll-out, roll-out post implementation validation, roll-out release, and operation. They propose that a quality gate be implemented after the conclusion of one phase and before the initiation of the next phase. They not only list some possible deliverables and responsible parties at each gate but also identify the criteria for the gates and sample metrics for the criteria. Table 3 provides sample Quality Gate criteria for the "build" phase in systems development lifecycle (SDLC) as proposed by Tarrani and Zarate [24]. Similarly, Farrell-Vinay [19] identifies several Quality Gates throughout the SDLC including requirements review, architectural design review, code review, system test, and deployment. In addition, Salger et al. (2009) focus on the design phase of information systems development and present detailed Quality Gates for software specification. Chenal and Schwartz [4] provide some practical guidelines for improving the software development process. One of these guidelines is that Quality Gates and criteria should be defined at the beginning of the unit testing phase, the integration testing phase, the system testing phase, and the user acceptance testing phase.

Furthermore, Quality Gates have been adopted by organizations such as Fidelity International, United States Department Veteran Affairs, and Microsoft Visual Studio Team for quality assurance purposes. For example, Fidelity International reported that six Quality Gates (pre-project, business study, functional model/design-build iterative, system integration testing and user acceptance testing, implementation, and post-project) were used with its agile software development method. What is interesting about these Quality Gates is that they consist of multiple criteria with a pre-determined weighting scheme and that the quality score at each gate is calculated based on the weighted sum of individual criterion scores.[1] Although this weighted calculation approach for quality scores may require more effort in the initial identification of all possible quality criteria, it allows project managers and other responsible personnel to adjust the weightings of certain criteria based on the specific nature of each project.

[1] Available at http://www.cmminews.com/2006/Pres2006/25th/Practitioner/Quality%20Assurance%20in%20an%20Agile%20Delivery%20Method.pdf

Table 3: Sample Quality Gate for the Build Phase of SDLC (adapted from [24])

Responsible	Deliverables	Quality Gate (Conditions to be met/ deliverable characteristics)	SQA Metrics (Defects Injected in Phase)
Developers Technical Writers Configuration Manager (Software and documentatio n control)	• Code(Source code, compiler object code and linked object modules, executable code, objects and classes) • DDL (Data Definition Language) scripts • Database Creation • DML scripts (Data Manipulation Scripts – Queries, etc.) ;(SQL, SQL* Plus, PL/SQL, etc.) • Stored procedures, triggers and encoded business rules • Server side scripts (csh, pearl, cgi, etc.) • Data dictionary • Data documentation (keys, constraints, E-R diagrams, triggers and stored procedures) • Technical documentation • Operational documentation • User documentation • Findings from peer, preliminary and critical reviews of all documentation • Library control logs (manual or automated) • Build analysis • Release notes Can also include: • Object request broker configuration • Access control list development	• Executable code passes unit and integration tests • Schema verification against database design documentation (includes validation of DDL scripts) • Data dictionary matches schema • Data dictionary, schema, and data documentation validated against one another • Stored procedures and triggers return expected results • DML and server side scripts validated • All documentation changes are traceable to source of/reason for change • All documentation matches system as built • Source code changes (including code, DDL/DML and server side scripts) can be traced to source of/reason for change • Code base is reconciled with library control logs • A build analysis is provided with the product • Release notes are provided with the product	• # unit test failures • # integration test failures • # build discrepancies • # instances of script rework • # discrepancies found during scheme verification • # discrepancies found in schema documentation (traceability, factual errors, instances of missing information) • Documentation follows FSS standards (Yes/No) • # source code reconciliation failures between code base and library control logs • # errors in build analysis • # errors in release notes (missing, incomplete or inaccurate information)

As another example of applying Quality Gates to software development, a team working on Microsoft Visual Studio has also adopted Quality Gates to improve software quality [18]. Before a feature can be checked into and integrated into the main product, it must go through a set of Quality Gates to ensure the quality and stability of the final product. These Quality Gates may include static code analysis, code coverage, localization testing, security implication analysis, API reviews, and so on.

Overall, the existing literature has identified that Quality Gates can be applied throughout the software development life cycle or within the development and testing domains. Yet, most studies so far have primarily focused on quality gates within the scope of a project. As such, we propose in this study that quality gates can be adopted as a mechanism to achieve better quality assurance throughout the enterprise IT production process. In the following section, we present a case study in the context of a large organization in the process of implementing quality gates.

CASE STUDY OF IMPLEMENTING QUALITY GATES TO SOFTWARE DEVELOPMENT LIFE CYCLE

FedEx is a Fortune 500 company with a diverse and complex software application portfolio that consists of customer facing applications, middleware applications, and backend systems that need to operate in concert in order to provide the best customer experience. Although the organization utilizes existing technologies and some commercial software products, a significant portion of the applications are developed in-house following a standardized software development process. Due to the complex nature of this portfolio (e.g., number, type, and interdependency of applications) and the composition of the software development teams (e.g., size, skill set, and process maturity), software quality assurance is especially critical to corporate revenues and customer satisfaction. As a part of the software quality assurance initiative, alternative approaches were considered to resolve the inconsistent levels of the quality of the code coming from disparate development groups, which are housed in multiple operational divisions.

In July of 2008, the enterprise testing organization within FedEx was renamed to Software Quality Assurance, with an expanded charter to provide processes and measurements to ensure software quality. Soon after, several quality improvement efforts (e.g., QA Prescripts, enterprise-wide IT renewal, improvements to the Global Development Process) were undertaken to develop key quality standards throughout the software development lifecycle, to improve standardization of development and testing deliverables, and to improve metrics around test coverage of requirements.

Subsequently, a special task team was set up to investigate the potential of Quality Gates as an approach to promoting software quality. The team consisted of an external consultant, vendor team and staff members from various areas in FedEx including Software Quality Assurance, Information Security, Revenue Systems, eCommerce, and Enterprise Architecture.

A FRAMEWORK OF QUALITY GATES FOR ENTERPRISE IT PRODUCTION PROCESS

In collaboration with a team from FedEx Software Quality Assurance and based on an extensive review of scholarly and practical studies, a framework for implementing quality gates for enterprise IT production process was developed as in Figure 2.

Like most Quality Gates research in software development, the framework captures the idea that Quality gates can be implemented as gates between software development phases (e.g., scope, build, and test). The framework characterizes these lowest-level quality gates as Heartbeat-level Quality Gates [16]. In general these gates combined with objective and metric-based criteria most often result in a pass or fail outcome. Typically these heartbeat-level metrics are gathered by automated tools. Some tools analyze the artifact itself only (e.g., syntactical analysis) whereas other tools examine the artifact in combination with other supplemental code that exercises the artifact (e.g., passing or failing or unit tests, code coverage).

From an intra-system perspective, these heartbeat-level gates are implemented within the production process for different disciplines (e.g., physical infrastructure and services, back-end/data systems, middle ware, end-user applications) that are involved with producing a software system. For example, physical infrastructure and services might go through a scope, build, and test production process to ready the physical environment for downstream disciplines. Between each of these phases, specific low-level quality gates may be implemented to ascertain the quality of the software components in meeting its pre-determined objectives. Subsequently, other disciplines (e.g., back-end/data systems, middle ware, end-user applications) will follow a similar pattern of implementing discipline-specific quality gates between the same phases.

At the end of each discipline's production process for the particular component, another quality gate is implemented to ensure that the quality is adequate for downstream disciplines to utilize. Although the production process for each discipline typically requires the completion of upstream discipline activities to be completed before finalizing its product, often some portion of the process may occur simultaneously. When the application production process is complete, which necessitates the artifact produced by other disciplines to have passed through intra-discipline and inter-discipline gates, the products have to pass through another quality gate before beginning systems testing.

Unlike most existing work related to Quality Gates, the framework also suggests that strategic-level Quality Gates should be implemented between layers of the enterprise IT production process. Strategic-level Quality Gates improve the ability to assess the current state of IT systems' quality through increased transparency and allow management to make more accurate risk-based decisions. While many of the decisions made at higher-levels of the software process (e.g., whether to include specific system changes in a particular project, whether or not a particular project should be released) still incorporate subjective evaluations of information streams, strategic-level Quality Gates attempt to reduce the subjectivity by introducing further structure/criteria to the decision making process. Furthermore, strategic-level Quality Gates often utilize abstract heart-beat level Quality Gate outcomes for use in the decision process thus increasing the objectivity of the information used in the decision making process. Different levels of management are focused on different boundaries and hence Quality Gates should be implemented between various layers of the overall enterprise IT production process including system-project, project-release, release and IT, and the alignment between IT and business.

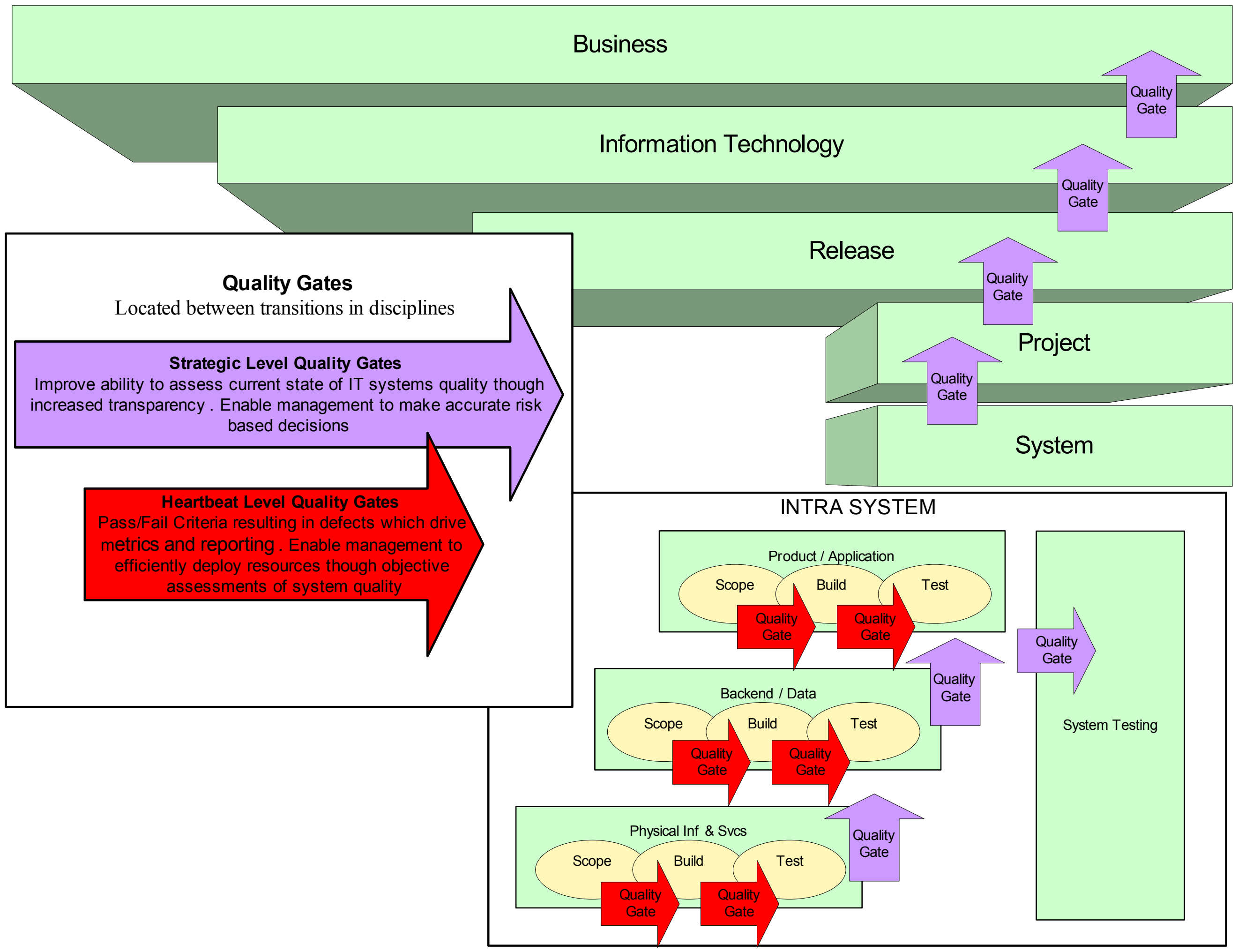
Business
Information Technology
Release
Project
System
Quality Gate
Quality Gates
Located between transitions in disciplines
Strategic Level Quality Gates
Improve ability to assess current state of IT systems quality though increased transparency . Enable management to make accurate risk based decisions
Heartbeat Level Quality Gates
Pass/Fail Criteria resulting in defects which drive metrics and reporting . Enable management to efficiently deploy resources though objective assessments of system quality
INTRA SYSTEM
Product / Application
Backend / Data
Physical Inf & Svcs
Scope
Build
Test
System Testing

Figure 2: A Framework for Quality Gates for Enterprise IT Production Process

INITIAL IMPLEMENTATION OF THE FRAMEWORK

The focus of the initial Quality Gates effort at FedEx was placed on intra-system heartbeat level quality gates. Automated static code analysis and code coverage analysis have been implemented during the build and test phases to improve code quality upon entry into the system testing phase.

The objective of performing static code analysis using an automated tool is to detect problematic programming practices and potential errors without actual execution during the development stage. Static code analysis has the potential to: 1) decrease production defects that are undetectable during black box unit testing and system testing, 2) increase productivity and code consistency due to adoption of good programming practices among large development teams, and 3) comply with internal or regulatory software quality initiatives. An open source software product for the static code analysis was adopted by the quality gates team to perform static code analysis using eleven customized rules sets (e.g., basic rule set, design rule set, optimization rule set) that are most applicable to the organization. This code analysis tool is capable of identifying potential problems in code such as suboptimal code, duplicated code, overcomplicated expressions, etc. Based on reports generated by code scans of this code analysis tool, developers now update their source code to ensure that all violations have been corrected.

The objective of performing code coverage analysis is to determine a quantitative measure of the percentage of program statements, branches, or paths that are executed by test cases in unit testing, integration testing, and system testing. A code coverage analyzer software was adopted to automate this process and the code coverage quality gate criterion is 90% unit test code coverage using structured testing/basis path testing.

These two quality gates are implemented between the build and test phases of the application discipline as an initial step in identifying and specifying quality gates throughout the enterprise IT production process. In addition, a process is under way to develop plans to implement heartbeat level quality gates for the other disciplines within the intra-system perspective. The next stage of implementing quality gates may focus on requirements documents (semantic scanning and requirements coverage), interfaces, or segmented testing. Additionally, implementation of strategic level quality gates throughout the organization will be investigated in more detail.

CONCLUSION AND FUTURE RESEARCH

Quality Gates have been used in variety of industries in order to increase quality of the artifacts that are the result of a production process. Quality Gates, which are normally implemented as entry and exit criteria between phases of a serial process, have been explored for use in software development. In this article we presented an overview of how Quality Gates have previously been applied and then proposed a framework of applying quality gates for enterprise IT production process.

Theoretically, building upon previous studies that have traditionally examined quality gates during the software development life cycle, this study proposes a holistic framework of implementing quality gates throughout the enterprise IT production process. In addition, we distinguish among quality gates at various levels such as intra-system level, project level, system level, release level, etc. Practically, the proposed framework provides guidance to organizations interested in implementing quality gates for software quality assurance.

Future research may focus on the quality gates implementation process at various levels of the software production process. It would also be interesting to explore how quality metrics obtained from heartbeat level quality gates can be used as criteria in strategic level quality gates. Another direction for our future research is to empirically investigate the organizational impact of quality gates implementation.

REFERENCES

[1] Brandon, D., *Project Management for Modern Information Systems*, IRM Press, 2006.

[2] Chang, K., Jackson, J., and Grover, V. "E-Commerce and Corporate Strategy: An Executive Perspective," *Information & Management*, Volume 40, 2003, pp. 663–675.

[3] Charvat, J. P. "How to Use Quality Gates to Guide IT Projects", http://articles.techrepublic.com.com/5100-10878_11-1061893.html, Feb 2010.

[4] Chenal, D., and Schwartz, P. "Improve Your Software Development Lifecycle Process – Practical Tips and Guidelines," StraightForward Tools and QAVantage, http://qavantage.toolsforproductmanagement.com/index_files/Improve_SDLC_Processes.pdf, Nov 2009.

[5] Cooper, R. G. "Stage-Gate Systems: A New Tool for Managing New Products - Conceptual and Operational Model," *Business Horizons*, May-June 1990.

[6] Dustin, E., *Implementing Automated Software Testing*, Pearson Education/Addison Wesley, 2009.

[7] Farrell-Vinay, P., *Manage Software Testing*, Auerbach Publications, 2008.

[8] Filho, W. P. "Quality Gates in Use-Case Driven Development," Synergia Systems and Software Engineering Laboratory Computer Science Dept. Federal University of Minas Gerais, Brasil, 1999.

[9] Flohr, T. “Defining Suitable Criteria for Quality Gates,” Lecture Notes in Computer Science; Vol. 5338, *Proceedings of the International Conferences on Software Process and Product Measurement*, 2008, pp. 245-256.

[10] Giebel, M., Essmann, H., Du Preez, N., and Jochem, R. “Improved Innovation through the Integration of Quality Gates into the Enterprise and Product Lifecycle Roadmaps,” *CIRP Journal of Manufacturing Science and Technology,* Volume 1, 2009, pp. 199–205.

[11] Hawlitzky, N. “ Integriertes Qualitätscontrolling von Unternehmensprozessen.Gestaltung eines Quality Gate-Konzeptes,” TCW Transfer-Centrum, München, 2002.

[12] Karlstrom, D., and Runeson, P. “Integrating Agile Software Development into Stage-Gate Managed Product Development,” *Empirical Software Engineering*, Volume 11, 2006, pp. 203-225.

[13] Nguyen, T. N., “A Decision Model for Managing Software Development Projects,” *Information and Management,* Volume 43, Number 1, 2006, pp. 63-75.

[14] Perry, W. E., *Effective Methods for Software Testing*, Wiley, 2006.

[15] Prefi, T. “Qualitätsorientierte Unternehmensführung, P3– Ingenieurges für Management und Organisation Verlag, ” Aachen, 2003.

[16] Rautiainen, K. “Cycles of Control: A Temporal Pacing Framework for Software Product Development Management,” Thesis, Department of Computer Science and Engineering, Helsinki University of Technology, 2004.

[17] Rothman, J. “How to Use Inch-Pebbles When You Think You Can’t,” Rothman Consulting Group, Inc. 1999, http://www.jrothman.com/Papers/Howinch-pebbles.html

[18] Saad, S. “Microsoft Visual Studio Team System Team Foundation Server: How We Use it at Microsoft,” Microsoft Professional Developers Conference, http://mschnlnine.vo.llnwd.net/d1/pdc08/PPTX/TL04.pptx, Feb 2010.

[19] Salger, F., Sauer, S., and Engels, G. “An Integrated Quality Assurance Framework for Specifying Business Information Systems,” *Proceedings of CAiSE Forum*, Amsterdam, The Netherlands, 2009.

[20] Scharer, M. “Quality Gate Approach with Integrated Risk Management: Methodology and Regulatory Guide for Target Oriented Planning and Processing of Production Procedures,” (Quality Gate-Ansatz mit integriertem Risikomanagement. Methodik und Leitfaden zur zielorientierten Planung und Durchführung von Produktentstehungsprozessen), Ph.D. Thesis, Karlsruhe Univ. (T.H.) (Germany). Fakultaet für Maschinenbau, 2002.

[21] Schmitt, R., and Scharrenberg, C . “Approach for the Systematic Implementation of Quality Gates for the Planning and Control of Complex Production Chains, ” *VIMation Journal Knowledge, Service & Production: IT as an Enabler*, Issue 1, 2008.

[22] Schmitt, R. “Governing the Process Chain of Product Development with an Enhanced Quality Gate Approach,” *CIRP Journal of Manufacturing Science and Technology* 1, 2009, pp. 206–211.

[23] Schneider, R. "Quality Gates: A New Device for Evaluation in Cross-Lingual Information Retrieval". In *Proceedings of the LREC-2004 Workshop of Transparency and Integration in Cross-Lingual Information Retrieval,* Lissabon, Portugal, May 30, 2004.

[24] Tarrani, M. and Zarate, L. “Life cycle Quality gates”, http://www.tarrani.net/LifeCycleQualityGatesZT.pdf, Feb 2010

[25] Valeri, S. G., and Rozenfeld, H. “Improving the Flexibility of New Product Development (NPD) Through a New Quality Gate Approach,” Advanced Manufacturing Nucleus, University of São Paulo, São Carlos, São Paulo, Brazil, 2004.

[26] Wallin, C., Ekdahl, F., and Larsson, S. “Integrating Business and Software Development Models,” *IEEE Software*, Volume 19, Number 6, 2002, pp. 28-33.

[27] Younack, R. “Quality Gates in Solution Projects,” RCG Information Technology, http://www.rcgit.com/rcg/uploads/File/WP-Q-Gates_in_Solution_Projects.pdf, Feb 2010.

ACKNOWLEDGEMENTS

We would like to acknowledge the contribution of Bart Dahmer, Rick Prendergast and other personnel in the Software Quality Assurance team at FedEx as well as personnel in other departments who contributed to the case study.

AUTHOR BIOGRAPHIES

Vladimir Ambartsoumian is currently pursuing his Ph.D. in the Department of Management Information Systems at the University of Memphis. He holds Master’s degree in International Business Administration,

University of Memphis, 2009 and Master of Science, Khakiv Polytechnic University, 1987. His research interests involve IT Enabled Co-Creation, Knowledge Management Systems, Business Value of IT, Data Mining, and Software Testing Processes.

Jasbir Dhaliwal is Professor of Information Systems and Associate Dean for Research and Academic Programs of the Fogelman College of Business and Economics at the University of Memphis. He is also a director of the Systems Testing Excellence Program at the FedEx Institute of Technology in the University of Memphis. He has a Ph.D. from the University of British Columbia, Canada and has published over fifty research papers in journals such as *Information Systems Research*, *Information & Management*, *IEEE Transactions on Engineering Management*, *International Journal of Production Economics*, *European Journal of Information Systems*, and in numerous conference proceedings of IS and related areas.

Euntae "Ted" Lee is currently an Associate Professor in the Department of Management Information Systems at the University of Memphis. He received his Ph.D. in Management Information Systems from University of Nebraska, MS in Computer Science from The Pennsylvania State University, and BS in Atmospheric Science from Seoul National University. His primary research interests are knowledge engineering and management, business rule managements systems, database systems, software development and testing, ethical decision-making in IT use and strategic use of organizational information systems. His work has been published in various journals and conference proceedings in IS and other related areas.

Thomas Meservy is an Assistant Professor of Management Information Systems at the University of Memphis. He graduated with his PhD in Management from the University of Arizona in 2007. Dr. Meservy graduated from Brigham Young University with a B.S. in Management and a Masters of Information Systems Management. His research interests include software development tools and methodologies, collaboration, and automated understanding of human nonverbal behavior. His work has been published in various leading journals such as *Data Base*, *IEEE Computer, IEEE Intelligent Systems, IEEE Transactions on Intelligent Transportation Systems, Group Decision and Negotiation,* and numerous proceedings of major IS conferences such as *ICIS*, *HICSS*, and *AMCIS*. Much of his research has been funded. Dr. Meservy also holds a number of professional technical certifications.

Chen Zhang is an Assistant Professor of Management Information Systems at the Fogelman College of Business & Economics at the University of Memphis. She has a Ph.D. in the area of Management Information Systems from Purdue University. Dr. Zhang's research interests include open source software development, user-driven innovation, information value, information behavior, and software development and testing. She has published her research in *Information Systems Research*, *AI Magazine*, and in proceedings of major international conferences such as *International Conference on Information Systems* (ICIS).