Quality Assurance Tradeoff Analysis Method (QATAM)
An Empirical Quality Assurance Planning and Evaluation Framework

Stefan Biffl[a] Christian Denger[b] Frank Elberzhager[b] Dietmar Winkler[a]

[a]TU Wien, Institute of Software Technology and Interactive Systems
[b]Fraunhofer Institute of Experimental Software Engineering (IESE)

ABSTRACT
The selection and design of quality assurance (QA) methods for software development projects a) involves tradeoffs that are not always made explicit and b) the impacts of the selection decisions on project success and risks may be not well understood. Similar to SEI's ATAM analysis technique for software architecture quality and risk evaluation, this paper introduces the ideas for QATAM, a technique for the evaluation of QA strategies and their tradeoffs in the context of a software process. In a framework process to define and evaluate software engineering and QA strategies, QATAM draws on approaches to elicit stakeholder value propositions and risks, to operationalize the most important value propositions (quality requirements) in scenarios, and to rate the potential of QA approaches to identify and mitigate relevant project risks. We illustrate the application of QATAM in an ongoing research project LifeCycleQM, which aims at improving evidence-based application of QA activities in SMEs.

Categories and Subject Descriptors
D.2.8 Software Engineering Metrics; D.2.9 Management Software Quality Assurance.

General Terms
Management, Experimentation, Measurement.

Keywords
Quality Assurance, evidence-based method selection, focusing empirical study planning, measurement planning.

1. INTRODUCTION
The quality of software development depends on sufficiently understanding the effects of software engineering approaches and quality assurance (QA) activities on product and project characteristics. Many empirical reports address the effects of the application of single methods in an empirical context; in real life decision makers need to assess and compare the overall effects of QA method combinations and the tradeoffs between involved QA activities. For example, directing resources towards methods that can effectively address some key risks in the project may be more beneficial than using general-purpose QA methods (see [1] and [12] for an overview). On the other hand, concentrating resources on a few risk items [13][14] may leave other risks unchecked that need at least to be observed (i.e., measured) in the course of the project.

A goal of the research project LifeCycleQM [5][6] is to support project managers and quality managers in small- and medium-sized enterprises (SMEs) in selecting, based on empirical evidence, QA approaches that effectively and efficiently help fulfilling desired quality goals and thus reduce project risks. Part of this ongoing work is the development of QATAM, similar to the ATAM approach [10] in software architecture evaluation and modeling. The purpose of QATAM is to assess the likely consequences of QA planning decisions in software projects based on empirical evidence.

This paper introduces the QATAM ideas and concepts, motivates expected benefits from applying the concept in research examples, and discusses issues that need further research work. Similar to ATAM, the steps in the QATAM framework elicit stakeholder win conditions (goals) and concrete scenarios to define project benefits and risks; identify risks that should be measured and/or mitigated with QA activities; find and evaluate the likely effects of QA activities (and their tradeoffs) in the project context using evaluation models that can range from informal to formal quantitative models and typically involve a range of empirical evidence.

The remainder of this paper is structured as follows: Section 2 summarizes the ATAM approach for scenario-based tradeoff elicitation and analysis; Section 3 introduces the steps in the QATAM concept; Section 4 provides examples that illustrate the approach and raise issues for further investigation; Section 5 concludes expected contributions of the approach.

2. QUALITY ANALYSIS WITH ATAM
The *Architecture Tradeoff Analysis Method* (ATAM) [10] is based on the notion that software architecture design involves tradeoffs, which are not always made explicit, and the impacts of design decisions may not be well understood. ATAM is a scenario-based and model-based analysis technique to analyze software architectures (with respect to multiple quality attributes) and explicitly consider design tradeoffs. The method aims at illuminating risks in the architecture through the identification of attribute trends early in the software development lifecycle.

Kazman et al. [10] list 9 ATAM process steps: The initial step 0 (p*lanning/information exchange*) includes a brief method presentation for all stakeholders and an initial overview of the proposed architecture, main quality goals, and an initial set of scenarios. The ATAM process starts at step 1 (*scenario brainstorming*). Key stakeholders collect important system scenarios, system defects and anticipated changes of the system. Scenarios operationalize software quality attributes in a project context, elicit context parameters, and performance measures; i.e. how can an attribute be measured, how much performance is enough, and how do characteristics of the architecture affect the observable manifestation. An observation from applying the method is that a focus on optimizing quality attributes in isolation may compromise the sufficient fulfillment of other, also important, quality attributes. These identified scenarios are mapped to a detailed architectural presentation in step 2 (*architectural presentation*) to achieve scenario coverage with all important attributes of the related scenario. This *scenario coverage checking* (step 3) is based on a set of quality attributes related to the application domain, performed by the key stakeholders. Step 4 focuses on *scenario grouping and prioritization* regarding individual stakeholder roles and requirements. The software architect maps *high-quality attributes of the selected high-priority scenarios* (step 5) to the architectural implementation to see their realization within the architectural design and the design impact on the quality attributes; how is the response of the architecture to scenarios on e.g., system performance or security. Based on these findings, the analyst identifies sensitive scenarios and attributes in a *quality attribute-specific analysis*. Based on a model regarding every quality attribute, e.g., performance issues, and small modifications of input variables, *sensitivity points* (step 6*)* are identified. Clusters of sensitivity points are the basis for *trade-off identification* (step 7) for individual requirements and scenarios. Finally, step 8 summarizes the findings derived from the analysis phases to find architectural improvements and action plans. ATAM aims at providing a repeatable and transitionable risk mitigation method to find architecture problems and potential stakeholder conflicts early in the system lifecycle.

3. INITIAL QATAM FRAMEWORK

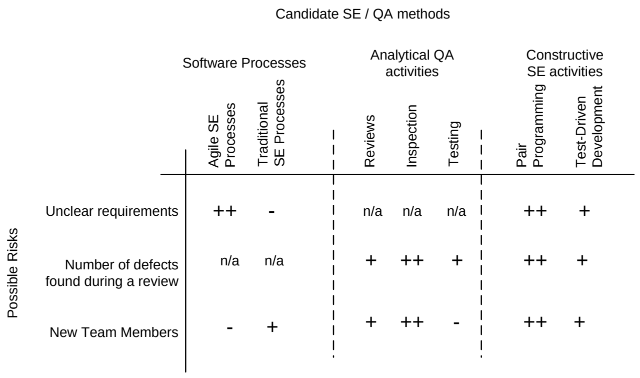
Similar to the well-established ATAM process we propose an initial framework for planning and evaluating QA activities. QATAM aims at supporting project and quality managers in planning appropriate QA activities along the software life-cycle. Depending on the application domain, well-defined methods and tools can support engineers in constructing high-quality software products. Nevertheless, it depends strongly on the project context to select a suitable set of methods to provide the stage for achieving high-quality software products. The initial QATAM framework consists of a sequence of steps similar to ATAM steps:

- *Step 0 (planning/information exchange)* establishes a common view on the project for all stakeholders taking part in the workshop. An important success factor in this step is to gather sufficient information for a clear picture of the project scope, context, and constraints.
- *Step 1 (scenario brainstorming)* identifies stakeholder win conditions, most important scenarios and measures for success criteria, i.e., "when is a product good enough from a quality point of view". EasyWinWin, a well-proven groupware supported approach, can be used for eliciting and prioritizing stakeholder win conditions [8][9].
- *Step 2* focuses on the initial selection of candidate QA activities and a basic set of quality measures. What QA activities are appropriate for investigating the products under the given context constraints? What level of quality is necessary to pass the initial quality gate? An example for a decision support framework for software inspection planning using empirical inspection knowledge can be found in [2], another framework example for planning QA techniques (i.e., inspection and testing) is described in [5].
- *Step 3* and *step 4* are similar to ATAM and cover (a) a scenario coverage checking procedure to focus on the most important business cases first and (b) a prioritization and grouping of scenarios, e.g., using techniques from the RiskIt process [11]; a range of project and quality complexity drivers can be found in CoCoMo II [3] and CoQualMo [4].
- An important difference between ATAM and QATAM is: ATAM evaluates product variants, while QATAM evaluates process/project variants, which needs to be reflected in the methods and evaluation results. In contrast to *ATAM step 5*, the QATAM process focuses on different variants of QA method sets, e.g., combinations of inspection and testing [1][12], different inspection reading technique variants with focus on defect types that were identified as important in usage scenarios. Method application knowledge can be derived from literature suggestions, experts, local (e.g., company) experience, and empirical studies.
- *Step 6 (sensitivity point analysis)* includes the comparison of different QA method set variants for determining their impact on important product quality attributes and measures. Depending on the available experience this may involve prototyping steps or empirical models such as CoQualMo.
- *Step 7* determines trade-offs between important quality attributes when variants of QA method sets are used in the development scenarios.
- *Step 8* summarizes candidates for most promising method sets and defines a detailed action plan. This action plan is based on the sensitivity point analysis and measures in the individual project context.

QATAM provides a decision framework for traceable and repeatable best-practice method planning and evaluation along the software process life-cycle. The expected benefits of QATAM come from support for project and quality managers regarding (a) defect detection for product cleanup, (b) product quality estimation approaches based on measurements and metrics (e.g., defect estimation), (c) exit criteria for strategic project decisions if defined goals at quality gates fail (e.g., demonstration laying the systematical foundation for some relevant future software capability with respect to functionality, performance, quality; contribution to project win conditions, etc.), and (d) elicitation of risk indicators, e.g., failure to demonstrate planned systematical achievement of future software capability. Finally, QATAM contributes to QA method improvement by providing systematic plans for measuring empirical data in real-world projects and well-defined contexts.

4. QATAM EXAMPLE APPLICATIONS

This section provides preliminary QATAM application examples (see also the Appendix). We basically distinguish between qualitative and quantitative methods for the evaluation of QA strategies in a given context, depending on the availability of local and/or general empirical data. Figure 1 depicts a snapshot from QATAM step 5, where workshop participants qualitatively rated the candidate methods in risks elicited from workshop scenarios. Unclear requirements, a high number of defects found during a review cycle, and new team members were identified as crucial for project success. Candidate software engineering and QA methods (e.g., software processes, analytical QA activities, and constructive software engineering activities) might help to mitigate these risks. Note, that an increasing number of "+" indicates a more suitable method to mitigate a risk (positive effect), while an increasing number of "-" describes a negative impact on risks. The evaluation of the techniques, e.g., "++" and "--", were based on the experience of the experts in the QATAM workshop.



	Candidate SE / QA methods						
	Software Processes		Analytical QA activities			Constructive SE activities	
Possible Risks	Agile SE Processes	Traditional SE Processes	Reviews	Inspection	Testing	Pair Programming	Test-Driven Development
Unclear requirements	++	-	n/a	n/a	n/a	++	+
Number of defects found during a review	n/a	n/a	+	++	+	++	+
New Team Members	-	+	+	++	-	++	+

Figure 1: Cut from a Risk – QA method candidate matrix.

For example, agile software processes have better potential to deal with unclear requirements better than waterfall processes. Software inspection was found to be a well suited approach for defect detection in early software products, even if requirements are unclear. Furthermore, software inspection is applicable as a learning framework for new team members. Also pair programming supports introducing new team member to an existing software development team. Summarizing the matrix in Figure 1, a combination of agile software processes, pair programming and an inspection approach might be an appropriate set of methods to reduce risks in the depicted situation.

Beside the qualitative evaluation it is also possible to support the trade-off analysis with empirical findings either gathered in a local context or initially from literature. For example, in case requirements defects are perceived as a high project risk, inspections proved to be an efficient means on detecting and removing these types of defects. However, there are many different variants on how to perform inspections, e.g., with or without a meeting, using checklists or scenario-based reading techniques, assigning different foci or the same foci to reviewers or involving more or less developers as reviewers. These are trade-off points of a single technique that need to be considered when defining the right strategy to resolve the project risk. For example, not performing a meeting reduces the overall time needed for an inspection but it also reduces the effect of team-learning. A quality manager must consider these trade-offs with respect to his or her goals – empirical data from literature can be used as a starting point for such quantitative analyses. However, in order to implement a sophisticated QA strategy it is important to measure and analyze data on the variation of defect detection effectiveness after the decisions are implemented. Furthermore, it is important to reason which QA activities really found certain types of defects in the project and which QA activities should have found these defects according to the QA project plan. An approach to fulfill such a measurement goal is described in [7]

5. CONCLUSION

We presented the ideas and concepts of a currently developed approach, QATAM, a framework for supporting quality managers and project leaders in defining and evaluating QA strategies in a certain development context. The framework takes well-working ideas from ATAM and provides a means to reason about potential implementations of QA and software engineering strategies based on the benefits, risks, and costs related to these strategies.

QATAM supports in early development stages the evaluation of potential benefits and risks of a range of software engineering and QA strategies (combination of software engineering and QA methods). The expected contributions of this approach are: 1. Repeatable scenario-based evaluation of capabilities of bundles of QA techniques (instead of single techniques in isolation) and tradeoffs between these techniques in a project context. 2. Document context parameters, rationale for QA method selection to enable measurement of relevant QA performance parameters within the project. 3. Use of best-available empirical evidence (local and/or from research literature) for QA method selection. Furthermore, applying QATAM will provide support in identifying gaps in empirical evidence that can drive local empirical data collection and help focus empirical research efforts.

Currently, researcher groups at TU Wien and the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern work jointly on the refinement of the QATAM approach and its application in the context of the research project LifeCycle QM funded by the BMBF grant 01 IS E05 D.

REFERENCES

[1] Aurum, A.; H. Petersson, C. Wohlin. State-of-the-Art: Software Inspections Turning 25 Years. Journal on Software Testing, Verification and Reliability, 12(3):133–154, 2002.

[2] Biffl S., Halling M. (2003) "Managing Software Inspection Knowledge for Decision Support of Inspection Planning", in *"Managing Software Engineering Knowledge"* edited by A. Aurum, R. Jeffery, C. Wohlin, M. Handzic, Springer Verlag.

[3] Boehm B.W., E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. W. Brown, S. Chulani, and C. Abts. *Software Cost Estimation with Cocomo II*. Prentice Hall, 2000.

[4] Chulani S. COQUALMO (COnstructive QUALity MOdel) —a software defect density prediction model. In *Proceedings of the ESCOM — SCOPE'99*, pages 297–306, 1999.

[5] Denger C., Elberzhager F.; Basic Concepts to Define a Customized Quality Assurance Strategy; IESE Report No. 013.07/E (Project LifeCycleQM); February 2007.

[6] Denger, C.; F. Elberzhager. A Comprehensive Framework for Customizing Quality Assurance Techniques, Report of WP 3.1 in the LifeCycleQM project, Report Number IESE-118.06/E, 2006.

[7] Freimut B., Denger C., Ketterer M. "An Industrial Case Study of Implementing and Validating Defect Classification for Process Improvement and Quality Management" 11th International Software Metrics Symposium. Metrics 2005 - Proceedings (2005)

[8] Grünbacher P., "Collaborative Requirements Negotiation with EasyWinWin", *11th International Workshop on Data-*

base and Expert Systems Applications, IEEE, London, 2000, pp. 954 - 960.

[9] Grünbacher P., Halling M., Biffl S., Kitapci H., Boehm B. W.(2004) "Integrating Collaborative Processes and Quality Assurance Techniques: An Example from Requirements Negotiation", Journal of Management Information Systems (JMIS), 20 (4): 9-30.

[10] Kazman R., Barbacci M., Klein M., Carriere S. J., Woods S. G., Experiences with Performing Architecture Tradeoff Analysis, Proc. ICSE 1999

[11] Kontio J., The Riskit Method for Software Risk Management, version 1.00, 1996. Computer Science TechnicalReports. Univ. of Maryland. College park, MD.

[12] Laitenberger O, DeBaud J-M (2000) An encompassing life cycle centric survey of software inspection. J Syst Softw 50(1):5–3

[13] Rus, I., Halling, M., Biffl S. (2003) "Supporting Decision-Making in Software Engineering with Process Simulation and Empirical Studies", Int. Jour. of Software Engineering and Knowledge Engineering (IJSEKE), 13 (5): 531-545.

[14] Winkler D., Varvaroi R., Goluch G., Biffl S.: An Empirical Study On Integrating Analytical Quality Assurance Into Pair Programming, Proc. ISESE06, Rio de Janeiro, Brazil, 2006.

Appendix

The following Figure A1 and Table A1 illustrate the strengths of candidate SE and QA activities in different parts of the software lifecycle. Figure A2 shows a QA strategy planning process for SMEs that embeds QATAM for evaluating bundles of QA strategies [6].

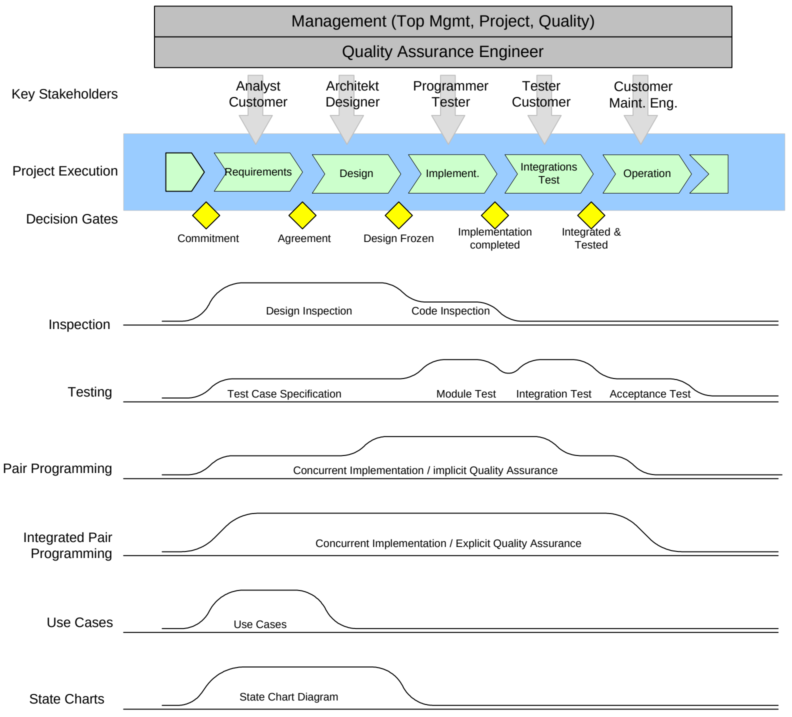


Fig. A1: Decision gates in the software life-cycle and phases when candidate methods can be applied.

Tab. A1: Decision gates in the software life-cycle and phases when candidate methods can be applied.

Method	Focus	Process step
Inspection ■ Design Inspection ■ Code Inspection	Requirements /Design Software Code	Early in the Life-Cycle Implementation Phase
Testing ■ Specification (Requirements) ■ Module Tests ■ Integration Test ■ Acceptance Test Exec.	Requirements /Design Code Composition of Modules Testing	Early in the Life-Cycle Implementation Phase Integration Phase Integration Phase / Operation
Pair Programming	Design / Implementation	Design/Implementation/Integration

Integrated Pair Prog.	Reqs/Design/Impl/Test	Full Life-Cycle Process
Use Cases	Requirements	Requirements Phase

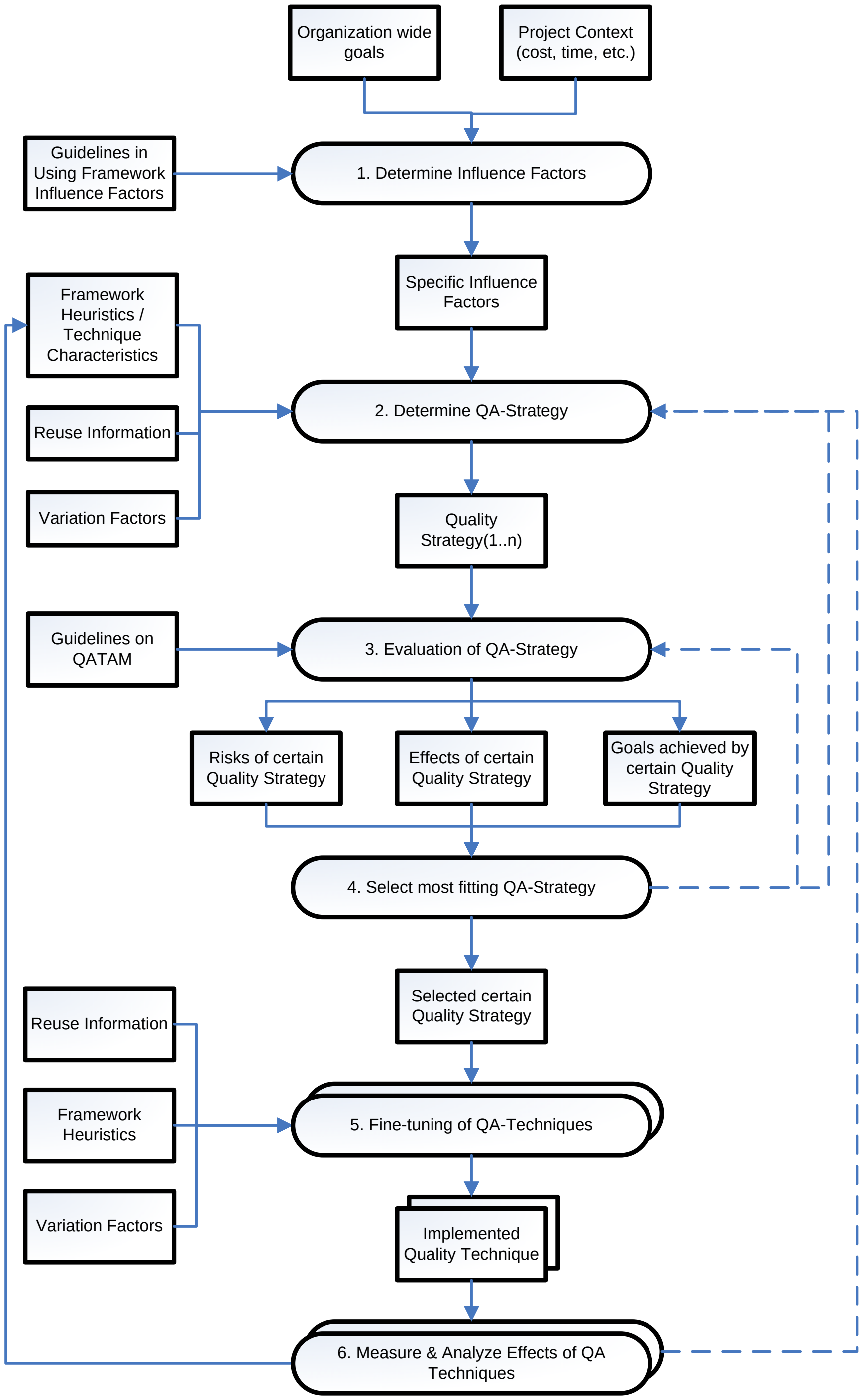

Organization wide goals
Project Context (cost, time, etc.)
Guidelines in Using Framework Influence Factors
1. Determine Influence Factors
Specific Influence Factors
Framework Heuristics / Technique Characteristics
Reuse Information
Variation Factors
2. Determine QA-Strategy
Quality Strategy(1..n)
Guidelines on QATAM
3. Evaluation of QA-Strategy
Risks of certain Quality Strategy
Effects of certain Quality Strategy
Goals achieved by certain Quality Strategy
4. Select most fitting QA-Strategy
Reuse Information
Selected certain Quality Strategy
Framework Heuristics
5. Fine-tuning of QA-Techniques
Variation Factors
Implemented Quality Technique
6. Measure & Analyze Effects of QA Techniques

Fig. A2: Exemplar overall planning process for developing a quality assurance strategy, developed in the LifeCycleQM project [6].