Applying a Selection Method to Choose Quality Attribute Techniques

Yin Kia Chiam[a,b,c,*], Mark Staples[a,b], Xin Ye[d], Liming Zhu[a,b]

[a]*NICTA, 13 Garden St, Eveleigh, NSW 2015, Australia*
[b]*School of Computer Science and Engineering, University of New South Wales, NSW 2052, Australia*
[c]*Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia*
[d]*Institute of Information and Decision Technology, Dalian University of Technology, Dalian, China*

Abstract

Context: Software products have requirements on various software quality attributes such as safety and performance. Development teams use various specific techniques to achieve these quality requirements. We call these "Quality Attribute Techniques" (QATs). QATs are used to identify, analyse and control potential product quality problems. Although QATs are widely used in practice, there is no systematic approach to represent, select, and integrate them in existing approaches to software process modelling and tailoring.
Objective: This research aims to provide a systematic approach to better select and integrate QATs into tailored software process models for projects that develop products with specific product quality requirements.
Method: A selection method is developed to support the choice of appropriate techniques for any quality attribute, across the lifecycle. The selection method is based on three perspectives: 1) risk management; 2) process integration; and 3) cost/benefit using Analytic Hierarchy Process (AHP). An industry case study is used to validate the feasibility and effectiveness of applying the selection method.
Results: The case study demonstrates that the selection method provides a more methodological and effective approach to choose QATs for projects that target a specific quality attribute, compared to the ad hoc selection performed by development teams.
Conclusion: The proposed selection method can be used to systematically choose QATs for projects to target specific product qualities throughout the software development lifecycle.

Keywords:
Quality Attribute Techniques, Technique Selection, Risk Management, AHP

1. Introduction

Acceptable levels of product qualities such as safety, performance, reliability and security are determined during software development. It is costly and time consuming to fix quality problems at later development stages if a system fails to meet the specified levels of product qualities. Poor software quality can lead to loss of life, personal injury, property damage, loss of money, damaged customer relations and business failure. It is therefore important to minimise the risks to poor product quality throughout the software development process. In this research, product-quality-risks (PQRs) are the potential problems or failures that may cause a system to fail to meet its specified levels of quality requirements. For instance, safety critical systems are concerned with software failures that may lead to hazards to life, property or the environment; security-critical systems aim to reduce faults that may lead to unauthorized access or control; and performance-critical systems aim to minimise the violation of constraints on response time or throughput.

Software-intensive systems such as medical systems and industrial automation systems use software to fully or partially develop most of their functions or requirements. The amount and complexity of the features provided by software have greatly grown over time. Therefore, the risks due to inadequate software product quality have become a major concern of development teams.

Development teams use a variety of specific techniques to identify, analyse, and control potential quality problems throughout the development of a system. In this research, we call these "Quality Attribute Techniques" (QATs). We define QATs as techniques, methods or practices used to identify, analyse, and control PQRs in the software and system development. These QATs are often technical engineering techniques [1] that address specific product quality attributes such as safety, reliability or performance. Examples of QATs for safety include hazard analysis techniques such as Failure Mode and Effect Analysis (FMEA) and Fault Tree Analysis (FTA). Examples of QATs for performance include Layered Queueing Network (LQN) and

*Corresponding author.
Email address: yinkia@um.edu.my (Yin Kia Chiam)

Stress Testing.

QATs may be specific to a single phase of the development lifecycle, or may span multiple phases. However, QATs are usually not explicitly listed in software process models, and the relationship between QATs and other process elements has not been well studied [1–3]. In order to create software process models that target specific quality attributes, it is important to first understand the important characteristics of QATs and how they relate to the development process.

Most software process tailoring methodologies are designed to address variations in organisation and project context such as size of the organisation, project or team, complexity of the project, and standard compliance [4–6]. As will be discussed in the next section, the research literature has not normally regarded product quality as an important characteristic of software process tailoring. In previous work [1, 2], we have argued that there is no systematic approach to represent and integrate QATs for arbitrary quality attributes within existing approaches to software process modelling and tailoring. Nonetheless, QATs are used in practice by software engineers. The existence of a repository of codified knowledge about QATs could help development teams to better understand the potential effect of using various QATs to target key product qualities across all phases of the software development process.

QATs can range from relatively simple and cheap walkthroughs and checklists, to more complex and expensive approaches requiring intensive expertise such as Fault Tree Analysis and formal methods. Development teams need to select appropriate QATs and incorporate them into development processes created or tailored for new projects. Most prior studies have focused on selecting QATs for a specific quality attribute (e.g. safety or performance) or for specific lifecycle phases (e.g. requirements, architecture). Some selection methods are only suitable for verification and validation QATs. Technique selection is an important part of the tailoring of technical development processes [2, 7]. The chosen techniques can significantly influence the software development processes and product quality.

The ultimate goal of this research is to help software development teams to improve product quality and quality assurance by helping them to better select and integrate QATs into tailored software process models that target specific product qualities. Our objective is to provide a better understanding of the relationship between QATs, development process models, and product qualities. In [3], we had proposed a framework to capture and present QAT information required for decision making during QAT selection and integration with development processes. The Quality Attribute Technique Framework (QAT Framework) provides a basis for creating a catalogue of QATs to support selection of QATs that target a specific product quality attribute. This QAT Framework includes a risk-management based categorisation scheme. QATs are categorised according to the method by which they address PQRs. We use risk management to understand how the QATs function to manage product quality by identifying, analysing, treating and monitoring PQRs.

In this paper, a quality-specific selection method is proposed, to help development teams choose appropriate QATs across the lifecycle. This is based on three perspectives: product-quality-risk management, process integration and cost/benefit. Risk management is used as a general theory to encompass a variety of product qualities. The multicriteria decision-making method (MCDM), Analytic Hierarchy Process (AHP)[8], is applied to analyse the cost/benefit of applying each candidate QAT. We sought to address the following research question: How can appropriate QATs be selected and integrated into process models that target a specific product quality? In an earlier paper [9], we presented an initial version of the QAT Selection Method, and evaluated it using an example real-world safety system taken from the literature. In the current paper, we present a revised selection method and further validation from an industry case study.

We report on a case study we have conducted to evaluate the method. In the case study, the QAT Selection Method was used to choose QATs that best fit the product quality requirements for a number of projects. The applicability of the QAT Selection Method was supported because the analysis indicated that only minor differences were found when comparing the actual selection made by the development teams. Compared to ad hoc selection by development teams, the QAT Selection Method provides a more systematic approach to provide explicit justification for the selection.

The remainder of this paper is organised as follows: Section 2 discusses work related to this research. Section 3 describes a brief background to the QATs, managing PQRs and the Analytic Hierarchy Process (AHP) that are used as the basis for our work. Section 4 describes the QAT Selection Method to choose QATs for projects that target a specific product quality. Section 5 presents a case study to validate the QAT Selection Method. Section 6 discusses the results and findings of the case study. Section 7 discusses the limitations and advantages of the QAT Selection Method arising from the evaluation. Section 8 presents conclusions and discusses future research.

2. Related Work

Desired product quality attributes can be achieved by using specific processes. Technical development processes can be created by incorporating into process models techniques that have an impact on specific quality attributes [7]. These processes can help development teams to manage product qualities.

Many researchers have addressed the need for selecting appropriate practices, process models, and techniques for a specific software project [10–15]. A wide range of techniques is available to choose from. These techniques have different quality impacts, benefits and limitations on their

use, and applied stages. Glass [13] emphasises that industry needs guidelines on how and when to use the various techniques in projects.

Safety standards such as IEC 61508 [16] and IEC 62279 [17] provide guidance on the selection of techniques and measures. According to the guidelines, appropriate techniques or measures are selected or omitted according to the target software safety integrity level. Recommendations (M: Mandatory; HR: Highly recommended; R: Recommended; –: No recommendation for or against being used; NR: Positively not recommended) are given to rank the appropriateness of the techniques and measures for different safety integrity levels. For example, when a technique is highly recommended, the development teams are expected to use the technique unless they can provide an argument that the technique would not reduce risks further, typically because equivalent outcomes can be achieved by using other techniques included in the software quality assurance plan. These standards aim to rank the appropriateness of techniques rather than provide a step-by-step approach to help development teams to choose a combination of techniques.

There are a number of approaches to help development teams to select techniques, methods and tools for software development. However, approaches such as [18–20] only target safety risk assessment techniques. Techniques for other product qualities are not considered. These approaches review and compare alternative hazard analysis techniques based on factors such as resources and constraints, and input and output requirements. The Zurich Risk Engineering approach [18] is more like a selection guide than a systematic approach. Bridges [19] and Lyon [20] have a more systematic approach but Bridges only focuses on safety hazard assessment, while Lyon focuses on the resources and information perspectives.

Perry [21], and Vegas and Basili [22] capture information about testing techniques and tools to aid in their selection. Perry [21] emphasizes that testing techniques and tools should be selected based on their ability to accomplish test objectives. He outlines a process of selection, first identifying the test factors, and then determining the testing objectives for each phase in the software development lifecycle. This approach only considers selection from the process perspective. Other perspectives such as cost/benefit and quality impacts are not included. Vegas and Basili [22] select the best-suited techniques for a given project based on a catalogue containing technique information. Desired values of attributes are compared with the values of each technique. Techniques that match the specified values are pre-selected and examined. Vegas and Basili [22] only focus on the selection of testing techniques.

Jiang et al. [23] explicitly link the attributes of a software project to the attributes of requirements engineering techniques, to help select requirements engineering techniques that are well suited to the project. This approach only considers selection for requirements engineering techniques based on project characteristics.

Various architecture evaluation methods are used to assess quality-related issues at the architecture level. Some methods conduct attribute-specific evaluations first and consolidate the results later [24]. These methods focus on reasoning models and expertise for the quality attribute required. Other approaches focus on the final stage of the decision making process to balance trade-offs and select the best candidates when there are conflicting quality requirements [25, 26]. Although these approaches consider quality aspects in their approaches, the methods only target selection for architecture design alternatives. These approaches focus on the design perspective.

Some studies [14, 18, 22, 27–29] characterize techniques to support their selection. Characterisation of techniques is intended to identify useful information that can help development teams to select techniques for their use in a software project [30, 31]. The chosen techniques can significantly influence the software development processes. Development teams are able to specify the processes and activities associated with the chosen techniques. Additional technical processes and activities can be added to the process to properly handle potential quality issues related to specific quality attributes [7].

In summary, prior work focuses on a specific quality attribute (e.g. safety) or lifecycle phase(s) (e.g. architecture, testing). Information which is important to process integration is missing in these approaches. To support the selection of techniques that target specific product qualities throughout the software development lifecycle, information about techniques must include aspects of product quality, process integration and cost/benefit.

3. Background

In this section, we give a brief background to QATs, managing PQRs and the Analytic Hierarchy Process (AHP) that are used as the basis for our work.

3.1. Quality Attribute Techniques Framework (QAT Framework)

In this research, QATs are the techniques, methods, or best practices that address the concerns or potential risks for specific quality attribute. Development teams apply various QATs to ensure that requirements of quality attributes are attained. There are a wide variety of QATs. For example, Jewell [32] states that performance risks can be reduced by applying proactive performance engineering techniques such as performance modelling, performance budgeting, and application profiling before performance testing starts.

In our earlier work [3], we proposed a QAT Framework to capture and present information about QATs to help development teams to understand QATs and to highlight their relationship to other process elements. A catalogue of QATs is intended to support development teams to select appropriate QATs and incorporate them into process

models and related process guides. The QAT Framework also aims to encompass QATs from many quality domains. QAT characteristics are considered from three perspectives: general information, process integration and selection. General information addresses how the QAT functions according to a risk-management based categorisation scheme. Process Integration describes the relationship of QATs with elements in software process models. For selection, characteristics capture costs, benefits, and quality impacts in terms of our risk-management based theory of quality management.

3.2. Managing Product-Quality-Risks (PQRs)

In our framework, QATs are classified according to risk management theory. Boehm [33, 34] and Charette [35, 36] discuss risk management and its importance in the software engineering context. Risk management can be applied in various contexts to meet different goals. There are several types of risk that can occur during a software development project (e.g. project risks, process risks, product risks and business risks). Appropriate techniques, methods and tools can be applied to analyse, avoid, reduce, minimise and eliminate the risks related to that software project. Some approaches apply risk management in the design and development of medical device software [37], [38]. Jones et al. [37] emphasizes that the risk management process must be an integral part of the quality management system. The concept of risk management can be extended to accommodate multiple quality attributes. Software quality related risks (e.g. safety, performance, reliability) need to be reduced to assure the quality of the software and hence contribute to the assurance of system quality [39, p.34].

In this research, our concern is to manage risks in relation to software product qualities across the software development life cycle. Numerous techniques can be used to assess and control PQRs. QATs are categorised according to risk management theory to help development teams select QATs through understanding how QATs function to manage product quality by identifying, analysing, treating and monitoring PQRs.

3.3. Analytic Hierarchy Process (AHP)

Numerous multicriteria decision making (MCDM) methods have been developed to help with decision problems by evaluating a set of candidates against pre-specified criteria. Examples of MCDM include Multi-Attribute Utility Theory (MAUT) [40], AHP [8], outranking techniques [41], weighting techniques [42], and fuzzy techniques [43]. In this research, we apply AHP to evaluate the candidate QATs according to cost/benefit criteria because AHP provides a convenient way to measure both quantitative and qualitative factors. AHP is widely used for many practical decision-making problems in industry and academia [8], for example in software requirements prioritisation [44] and software architecture evaluation [24–26]. These studies have reported the effectiveness of using AHP to set priorities and analyse tradeoffs and sensitivity.

The decision-making process in AHP [8] is based on relative assessment. In AHP, all candidates are evaluated using pairwise comparisons. As a result, the evaluation is less sensitive to judgment errors when compared to other MCDM methods using absolute assignments [25, p.246]. It is easier for decision makers to state their preferences if they focus on a small part of the analysed problem, i.e. only two chosen elements. The risk of an error is reduced, and the weights and grades determined in this way more accurately represent the user's actual priorities and preferences.

However, exhaustive pairwise comparison is time consuming when there are many alternatives to be considered [45]. To mitigate this problem in our research, QATs are clustered into smaller groups based on risk management and process integration perspectives. AHP is then applied to compare QATs from smaller groups of candidates. Some AHP analysis tools, such as Expert Choice and MakeItRational are available to compute the weights and grades of pairwise comparisons based on matrix algebra. In this research, MakeItRational [46] was used to perform calculations and analyse data.

4. QAT Selection Method

The QAT Selection Method described here is intended to help developers to choose QATs for any quality attribute, across the lifecycle. The method compares and evaluates QATs based on three perspectives: Product-Quality-Risk Management, Process Integration and Cost/Benefits.

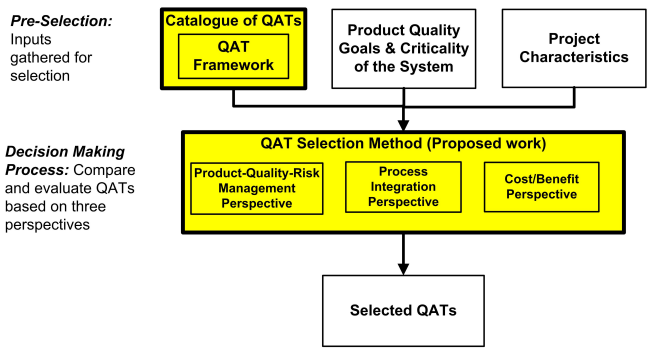

Figure 1: Overview of the QAT Selection Method (Our earlier work, the QAT Framework and our proposed work, the QAT Selection Method are highlighted in yellow. A catalogue of QATs has been developed based on the QAT Framework to provide the input, QAT information for the selection process.)

4.1. Pre-selection

In the pre-selection stage, QAT information and project information are gathered to provide inputs for the decision-making process. The following steps are performed:

- Gather the characteristics of the project, e.g. software process model used, team size, team expertise, standards used, degree of complexity of the software system, time constraints, cost constraints, and resource constraints.

- Determine the product quality goals/objectives and criticality of the system, e.g. the product quality attribute(s) that the software system intended to achieve.

- List the candidate QATs from which the selection is to be made. This list might be taken from an existing catalogue, or be extended with additional candidate QATs. An existing catalogue of QATs provides information about candidate QATs. This catalogue can be revised for adding new candidate QATs and capturing the characteristics of QATs using QAT Framework [47].

Taking into account the relevant QAT characteristics and project characteristics, suitable QATs are selected for the situation and to achieve the product quality goals. The selection process involves comparing and evaluating the characteristics of a list of candidate QATs. Development teams make the selection based not only on their personal knowledge but also on the relevant QAT information captured by the QAT Framework [47].

4.2. Decision-Making Process for Selecting QATs

In this stage, candidate QATs are compared and evaluated according to the three perspectives: Product-Quality-Risk Management, Process Integration and Cost/Benefit, to decide the best-suited QATs for the given projects. Risk Management and Process Integration perspectives will be used to form groups of QATs from which the selection is made. The shortlisted candidates will be evaluated using the AHP [8] method based on cost/benefit selection criteria. Safety QATs will be used as examples to illustrate the QAT Selection Method below.

4.2.1. Stage 1: Product-Quality-Risk Management Perspective

Risk management theory is used to compare candidate QATs based on how they address PQRs. Each QAT has different kinds of impacts, benefits and limitations in managing PQRs. The categorisation scheme is based on AS/NZS 4360: Risk Management (see Fig. 2). Following the categorization scheme, the candidate QATs from which the selection is to be made can be classified into two main categories: PQR assessment and PQR control. PQR assessment techniques are QATs that can be used to identify, analyse and evaluate the PQRs. PQR control techniques are either treatment QATs that can be used to avoid, reduce, minimise or eliminate the identified risks or monitoring techniques that can be used to determine whether the treatments are effective in addressing the identified risks

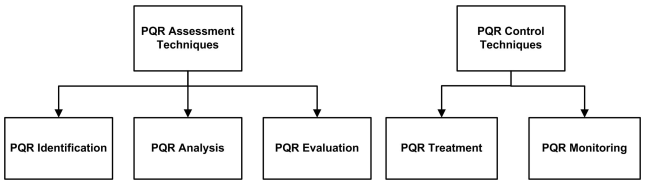

Figure 2: Categorisation scheme based on AS/NZS 4360:2004

as planned or required. The definition of each category can be found in [3].

This categorisation of QATs is intended to help development teams select QATs based on how they manage PQRs. QATs in the same category can be compared according to these factors, by referring to information in the QAT Framework [47]:

- type of results needed,

- characteristics of the system that are applicable (e.g. complexity of the risk, software safety integrity level),

- event of concern (e.g. single/multiple failure(s), human error), and

- type of application.

No single QAT can handle all aspects of system quality, especially for high assurance systems. If development teams apply only one or two QATs without understanding the function of the QATs in quality risk management, the intended product quality may not be achieved. A combination of QATs across the risk management lifecycle should be selected for effective PQR assessment and control across the software development lifecycle.

4.2.2. Stage 2: Process Integration Perspective

The Process Integration perspective is used to integrate candidate QATs into the software development lifecycle, and to cover the lifecycle. Some QATs have different impacts in managing PQRs when they are applied at different stages of the lifecycle. One way this can arise is when a QAT can be performed in more detail in later lifecycle phases. For example, Fault Tree Analysis (FTA) has limited usefulness when it is used to construct generic fault trees before design details are known. When the information of complete system design and a thorough understanding of the system are available, FTA can be applied to perform detailed analysis. Also, FTA can be used to completed or existing system to verify that the system is safe [48].

We have used AS/NZS ISO/IEC 12207 [49] as a basis for defining a generic software development process.

- Requirements Elicitation: "to gather, process, and track customer needs and requirements throughout the lifecycle so as to establish a requirements baseline that serves as the basis for defining the needed work products" [49, F.1.3.1, p.11].

- Software Requirements Analysis: "to establish the requirements of the software elements of the system" [49, F.1.3.4, p.13].
- Software Design: "to provide a design for the software that implements and can be verified against the requirements" [49, F.1.3.5, p.13].
- Software Construction (Code and Unit Test): "to produce executable software units that properly reflects the software design" [49, F.1.3.6, p.14].
- Software Integration: "to combine units into integrated software items, consistent with the software design, that demonstrate that the functional and non-functional software requirements are satisfied" [49, F.1.3.7, p.14].
- Software Testing: "to confirm that the integrated software product meets its defined requirements" [49, F.1.3.8, p.14].

Table 1 shows a mapping of QATs between their risk managment role and software development lifecycle phase. This table was derived from a study mapping safety QATs into the PQR management lifecycle and software development lifecycle [50]. The safety QATs were first mapped into corresponding software development lifecycle according to the recommendations in the literature. The safety QATs were fit into PQR management lifecycle by analysing their impact in addressing PQRs. The definition of each safety activity was compared with the definition of each PQR management activity. In safety engineering, the safety activities have the greatest degree of similarity in objectives lies in the PQR activities. This mapping study shows that the PQR management activities take place throughout the software lifecycle to assess and control PQRs.

After matching PQR management activities to the software processes, development teams select QATs which are intended to perform PQR management activities in specific phases. For example, QATs can be used to identify lists of PQRs for software systems during requirements elicitation and software requirement analysis. Development teams can use QATs to investigate the causes and consequences of identified PQRs, evaluate and prioritise the severity of the PQRs during software requirement analysis and software design phases. QATs to treat the PQRs (e.g. elimination of cause and event, reduction of the effects of negative consequences) can be selected and applied in software design and coding. Verification and validation QATs can be used throughout software development processes to monitor and review the product quality requirements, design and coding and to ensure the selected treatments address the PQRs.

Process information captured for QATs is intended to help development teams to select QATs for risk management activities and software phases. These QAT characteristics include inputs (the pre-requisite work products), outputs (the work products created or modified), and the development process phase(s). In each phase, different levels of detail of information are available for QATs. Some QATs generate output or results for other process activities or QATs. Appropriate QATs can be selected based on the types of inputs available and output required in each phase. Some QATs can be used in multiple development phases. In later phases, such as design and coding, the range of QATs available to be selected increases when more detail information about product quality requirements and design is available.

4.2.3. Stage 3: Cost/Benefit Perspective

AHP [8] is used to evaluate the ranking of each candidate QAT under cost/benefit selection criteria. An initial review of literature relevant to software safety (e.g. [18–20, 48, 51, 52], software architecture design (e.g. [25, 26]) and software testing technique (e.g. [27, 53]) identified criteria which influence the cost/benefit of applying a QAT. Criteria that can be used to analyse the tradeoffs between cost/benefit of applying each candidate QAT are as follows:

- Quality Impact: Impact of QATs in managing PQRs.
- Cost of Application: The effort, time and resource required.
- Ease of Use: The complexity of applying the QAT.
- Expertise: The knowledge, experience or training required or available.
- Regulatory (standard) Requirements: Recommendations by regulator/standard.
- Contractual Requirements: Requirements specified by contract.
- Tool Support: Availability of the tool support.

AHP [8] involves five steps for evaluation of candidate QATs. We use an example to illustrate these steps. Four PQR analysis techniques for safety: HAZOP (Hazard and Operability Study), FMEA (Failure Modes and Effect Analysis), FTA (Fault Tree Analysis) and ETA (Event Tree Analysis) are used as candidates for selection in this example.

Step 1: Define the evaluation criteria used to select candidate QATs.

First, we need to decide the selection criteria used to compare and evaluate candidate QATs. In our example, we want to evaluate the four safety PQR analysis techniques based on four selection criteria: Cost of Application, Expertise, Ease of Use and Quality Impact.

Step 2: Weighting selection criteria using pairwise comparisons or direct ratings.

The relative importance of one criterion over another can be expressed by using pairwise comparisons or direct

Table 1: Mapping of Product-Quality-Risk Management Activities to Software Development Process

	Requirements Elicitation	Requirements Analysis	Design	Construction	Integration	Testing
PQR Identification	X	X	X	X		
PQR Analysis		X	X	X		
PQR Evaluation		X	X	X		
PQR Treatment		X	X	X		
PQR Monitoring	X	X	X	X	X	X

ratings. Each selection criteria may have varying degrees of importance, depending on the needs of the development teams and also the criticality of the software systems. Development teams can use their judgement to weight the relative meaning and importance of each criteria based on the QAT and project characteristics. The development teams can use direct ratings instead of pairwise comparisons if the weights of criteria are known. For pairwise comparisons, every pair of selection criteria is compared using the weighting scale (1=Equal importance; 3=Moderate importance; 5=Strong importance; 7=Very strong importance; 9=Extreme importance; 2,4,6,8=Intermediate values between two adjacent judgement) [8]. Once the pairwise comparisons matrix is formed, weights of criteria are calculated by solving the eigenvector of the pairwise comparison matrix. When pairwise comparison is too time consuming, direct ratings allow us to provide the relative weights for all the criteria directly. In our example, we use pairwise comparisons to weight the priority among four criteria: with quality impact as the most important with a weight of 50.83%, followed by expertise (26.53%), cost of application (15.12%) and ease of use (7.52%).

Step 3: Determine relative ranking of each candidate QAT over another under each criterion.

For each criterion, the values of the weighting scale in the Step 2 are used to weight the preference of each candidate over another in pairs. The information captured by the QAT Framework for each candidate is intended to help development teams to make these judgements with reference to the literature, practitioners and experts. In our example, pairwise comparisons were made between HAZOP, FMEA, FTA and ETA for each criterion.

Step 4: Compute the overall value score for each alternative with all criteria considered.

Finally, the overall value score for each candidate can be computed with all criteria considered. The judgments (pairwise comparisons) are transformed into weights (criteria) and utilities (candidates). The weights and utilities are calculated and transformed into candidates ranking. In our example, the overall value score and candidates ranking were analysed for HAZOP, FMEA, FTA and ETA using the analysis tool. Table 2 shows the value score of candidate QATs comparison for each criterion. The value scores are hypothetical to illustrate the application of AHP. Table 3 depicts the overall value score for candidate QATs ranking weighted with criteria priority. Fig. 3 illustrates a chart for candidates ranking weighted with criteria priority. In this example, FTA has the highest ranking, followed by HAZOP, FMEA and ETA.

Step 5: Perform sensitivity analysis.

Sensitivity analysis provides information on how the candidates' rankings behave in response to changes in priorities. This will help to analyse stability of candidates ranking. This analysis can be illustrated using diagrams. In our example, Fig.4 illustrates a sensitivity analysis diagram for criterion, Cost of Application. The current weight for this criterion is 15.12% and the current overall value score for HAZOP, FMEA, FTA and ETA are 29.13%, 21.04%, 31.12% and 18.71%. The diagram shows how the candidates ranking change in response to changes in weight priorities for this criterion, Cost of Application.

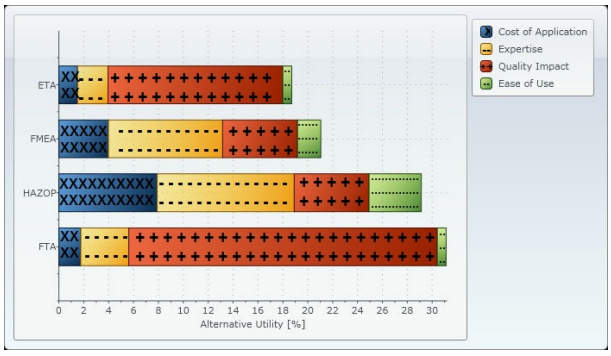

Figure 3: Candidates ranking weighted with criteria priority (illustrative example)

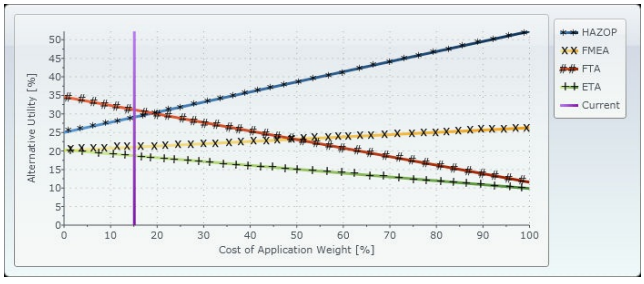

Figure 4: Sensitivity analysis diagram for criterion, Cost of Application (illustrative example)

4.3. Select QATs

Based on the comparison and analysis from these three perspectives, development teams can select a combination

Table 2: Candidates Comparison for Each Criterion (illustrative example)

Criterion	HAZOP	FMEA	FTA	ETA
Cost of Application	52.99%	26.26%	11.58%	9.88%
Ease of Use	55.96%	24.95%	9.55%	9.55%
Expertise	41.49%	34.64%	14.60%	9.27%
Quality Impact	11.82%	11.82%	48.74%	27.62%

Table 3: Overall Value Score for Candidates Ranking Weighted with Criteria Priority (illustrative example)

Candidate	Total	Cost of Application	Expertise	Quality Impact	Ease of Use
FTA	31.12%	1.75%	3.87%	24.78%	0.72%
HAZOP	29.13%	7.90%	11.01%	6.01%	4.21%
FMEA	21.04%	3.97%	9.19%	6.01%	1.88%
ETA	18.71%	1.49%	2.46%	14.04%	0.72%

of QATs to manage PQRs throughout software development lifecycle for their projects. Justifications for the selected QATs can be provided through qualitative and quantitative evaluation of candidate QATs according to these three perspectives. The candidate rankings analysed by AHP are used to assist the development teams to make the final decision. When making a selection decision for QATs in a category, the candidate QATs with high aggregated utility will be selected or shortlisted for final selection.

5. Case Study: Applying the QAT Selection Method

The objective of the case study was to evaluate the effectiveness of the QAT Selection Method. The QAT Selection Method was applied to retroactively choose safety QATs for three projects. Candidate QATs were compared and evaluated systematically using the QAT Selection Method.

A case study approach is particularly suitable for addressing "how-and-why" type research questions [54]. When there is a need to investigate the importance that context plays in relation to the subject being researched, the case study approach can be applied to study a specific bounded case or a number of cases in a particular setting. In this research, the case study approach was used to evaluate the proposed work. The methodology for this case study was based on the guidelines proposed by Runeson and Höst [55].

5.1. Study Context

The case study was conducted in Australia. It is an embedded case study in one company with three different projects as units of analysis. The context is the same company and same application domain. An embedded case study is a case study containing more than one sub-unit of analysis [54]. The company was selected opportunistically based on an existing academic-industry relationship. The company is an automation specialist for safety-critical infrastructure. The units of analysis were selected to fit the specific case study purposes.

Three projects were selected for three reasons. Firstly, the three development projects target a specific product quality, safety. Secondly, these three projects developed systems certified to different software safety integrity levels (SWSIL). Finally, development teams from these three projects applied different sets of QATs to target software safety. These projects are good for comparison because they are in a similar organisational and project context, but are good contrasts for our evaluation because they target different SWSILs, had different QAT selection criteria, and chose different combinations of QATs.

SWSIL is "a classification number which determines the techniques and measures that have to be applied in order to reduce residual software faults to an appropriate level" [17]. A SWSIL is one of the following five levels: 4 (very high), 3 (high), 2 (medium), 1 (low) and 0 (non-safety-related).

- Project 1 developed a safety real-time control system certified to SWSIL 3.
- Project 2 developed an information and real-time control system certified to SWSIL 2.
- Project 3 developed an offline design tool certified to SWSIL 2.

These projects were required to comply with the International Standard IEC 62279 for Railway applications (Communications, signalling and processing systems - Software for railway control and protection systems). There are 70 safety and performance QATs recommended in this standard. These candidate QATs were compared and evaluated using the proposed QAT Selection Method.

5.2. Data Sources

In this case study, a questionnaire was the major source of data, and a follow-up interview was conducted to collect general feedback and suggestions from the participants after completing the questionnaire. The selection criteria, project information and also the QAT information were required to choose the appropriate QATs for each project

using the proposed QAT Selection Method. The following preparation was completed before conducting the data collection.

5.2.1. Questionnaire

A questionnaire was designed to collect the data of the case study. The questionnaire comprises two sections. The first section was to collect information about individual participants and projects. The second section included six questions to collect data for each candidate QAT:

1. Which category matches how the technique functions to manage product-quality-risks?
2. Do you use this technique in the project?
3. Which ranking best fits your personal expertise/experience and team capability (Not at all, Low, Medium, High Very High)?
4. Which lifecycle phase(s) this technique had been applied to?
5. What were the reason(s) for selecting the technique?
6. What were the reason(s) for not selecting the technique?

We had included all the QATs suggested by IEC 62279 in the questionnaire. The participants were required to classify each safety QAT according to the risk-management based categorisation scheme. They could choose more than one category for how the QAT managed product-quality-risks. In the questionnaire, participants were allowed to add QATs not listed in the standard but which had been used in the projects. The questionnaire provided participants with reasons for selecting and not selecting QATs. These reasons were derived from factors influencing the selection of QATs. Participants were also allowed to add reasons not listed in the questionnaire.

5.2.2. Training Document

To assist the participants in correctly classifying the QATs, a training document which explains the risk management theory and risk-management based categorisation scheme was prepared and presented to participants immediately before the questionnaire. This training document was intended to help the participants understand the proposed risk-management based categorisation scheme. Additionally, a QAT catalogue was extracted from IEC 62279 and provided to participants, giving a brief description for each QAT.

5.2.3. Interview Questions

Interview questions were formulated based on the objective of the case study. The interview was conducted to collect general feedback and suggestions from the participants after filling in the questionnaire. The purpose of this interview was to justify whether the participants had the right understanding about the risk-management based categorisation scheme. This may affect the results of the questionnaire analysis. Also, we wanted to identify reasons for QAT selection that had not been asked in the questionnaire. These reasons can be considered as new selection criteria for our approach. The interview questions were:

- Do you have any general feedback about the risk-management based categorisation?
- Based on the presentation, are you clear about the concept of applying risk management theory in managing product-quality-risks?
- Are there any other reasons not listed in the questionnaire that would be relevant to the technique selection?

5.2.4. Capture QAT Information and Categorise the QATs

The QAT Framework [47] is used to capture information on safety and performance QATs from the literature. On a technique-by-technique basis, the relevant research papers, guidelines, books and electronic sources (e.g. publication available from websites) that describe or review a QAT in theory or practice were identified by the researchers. Information about the QAT characteristics proposed by the QAT Framework [47] was recorded. The QAT information provided input to compare and evaluate the QATs for further selection. Also, two researchers classified QATs independently, based on the risk-management based categorisation scheme. The categories assigned to each QAT were compared and finalised in a joint meeting.

5.2.5. Trial Case Study and Pilot

A trial case study was performed before conducting the actual case study. Five participants, including four software engineering Ph.D. students and one researcher were invited for this trial. The objective of this trial was to estimate the time required to complete the questionnaire and to collect the feedback regarding the design of questionnaire and the training material for the risk-management based categorisation scheme. Based on the feedback obtained from the trial participants, the following changes were made to improve the data collection:

- **Training document for risk-management based categorisation scheme:** To avoid misunderstandings between product-quality-risks and project-risks or business-risks, an example to describe the product-quality-risks management process was added. Also, a description of each category was elaborated with more details to improve the understanding of the categorisation scheme.
- **Presentation:** A presentation was given before the participants filled in the questionnaire. In the presentation, the risk-management based categorisation scheme was introduced. This helped to improve the

understanding of the categorisation especially for participants who did not have any background in risk management. The procedures to fill in the questionnaire were explained.

- **Questionnaire:** To reduce the time to complete the questionnaire and ease the data collection and analysis, the questionnaire was converted from Microsoft Word documents to Microsoft Access forms. The instructions in the questionnaire were revised (e.g. to clarify that the participants could choose multiple answers). In addition, a questionnaire guide was prepared. This helped participants to better understand each question and to enter the data correctly into the Microsoft Access forms. Some questions were revised to avoid misunderstanding.

5.3. Data Collection

Participants: The case study was conducted with three industry experts: 1 senior software engineer, 1 senior RAMS (reliability, availability, maintainability and safety) Engineer and 1 R&D RAMS Manager. The RAMS engineer and RAMS manager played important roles in managing and executing systems safety assessments. They were responsible to ensure that all the components in a system, including software met the required level of product quality for safety, availability, reliability and maintainability. The senior software engineer provided inputs related to software in system safety analysis to analyse and evaluate the potential safety risks in software components. He was also involved in deciding on and implementing the treatments appropriate to avoid, minimise or reduce safety risks. In these projects, the system safety analysis was done entirely within the project. An industry expert who had been actively involved in the selected project represented the development team of each project:

- Senior Software Engineer - Project 1
- Senior RAMS Engineer - Project 2
- R&D RAMS Manager - Project 3

The following documents were provided to participants before filling in the questionnaire: Training guide for the risk-management based categorisation scheme, questionnaire guide, QAT catalogue, participant information sheet and consent form. The procedure was as follows:

1. The questionnaire guide, training document, QAT bibliography, and participant information sheet and consent form were distributed to each participant.
2. A presentation was given to introduce the risk-management based categorisation scheme. An overview of the questionnaire in Microsoft Access forms was explained to assist the participant in understanding the questions and procedures.
3. Each participant used a laptop to fill in the questionnaire.
4. After completing the questionnaire, general feedback was obtained from each participant regarding the categorisation scheme through interview and a feedback form. Participants were allowed to write down their opinions on a feedback form.
5. A follow-up interview session was conducted to discuss the results of data analysis.

In this case study, data was collected mainly through a questionnaire. Each participant filled in the questionnaire for one project and for every QAT. Data collected include the QAT categories assigned by each expert that were used to provide inputs to evaluate the QAT Selection Method.

A semi-structured interview method was chosen, which supports this type of exploratory and validation study. Two researchers conducted the interviews together, the interviews were audio recorded, and later transcribed. The interviewers also allowed participants to write down their opinions on a feedback form provided.

5.4. Data Analysis

The following inputs were collected and analysed to validate the QAT Selection Method:

- Inputs from the questionnaire: project characteristics, list of QATs used or not used in the project (Question 2), the level of team expertise for each QAT (Question 3), lifecycle phase(s) that the QAT had been applied in practice (Question 4), reason for selecting and not selecting the QATs (Questions 5 and 6)
- Inputs from the QAT Framework [3]: Catalogues of safety and performance QATs which include QAT information and categories assigned to each QAT [50].

The inputs obtained from questionnaire were used to compare and evaluate the QATs based on the cost/benefit perspective. The selection criteria and weights assigned to each criterion in AHP [8] were based on the priority of the factors analysed from the collected data. Information about QATs was used to compare the QATs based on the PQR management perspective, process integration perspective and cost/benefit perspective. Finally, the QATs chosen for each project using the QAT Selection Method were compared with the actual list of QATs selected by the projects.

5.4.1. Pre-Selection

Before applying the QAT Selection Method, the following inputs were collected and analysed:

- The characteristics of the three projects as captured through the questionnaire (see Table 4).
- Candidates identified: 70 safety and performance techniques are listed in the standard. All the QATs are safety-related. In this content, four performance

techniques: Performance Modelling (PM), Performance Requirements (PR), Avalanche/Stress Testing (A/ST) and, Response Timing and Memory Constraints (RT&MC), had safety implications and were included in the comparison.

- The relevant characteristics for the candidate QATs were captured using the QAT Framework [3]. For each QAT, highly cited safety and performance research papers and text books were identified to describe or review the QAT in theory or practice. The information of safety and performance QATs was captured. The QAT information (aims, description, performer(s), lifecycle phase, input needed, output produced, guidelines available, provide output for other QAT, benefits, limitations, cost of application, expertise/knowledge required, team/individual approach, tool(s) support) were recorded from the literature. When there was uncertainty or lack of information, more literature (e.g. papers, guidelines, books and electronic sources) was searched to find the relevant information for these characteristics. Two catalogues of safety and performance QATs were developed based on the QAT information collected.

Safety is the main target quality attribute for the software systems developed by these three projects. Performance has less quality impact, especially for Project 3. For the purpose of this evaluation, the selection only chose QATs to target safety. The five goals of PQR management for this study were defined as follows:

1. Identify safety PQRs.
2. Analyse safety PQRs.
3. Evaluate safety PQRs.
4. Treat the identified safety PQRs.
5. Monitor the safety PQRs.

5.4.2. Stage 1: Compare Candidate QATs from Product-Quality-Risk Management Perspective

The candidate QATs listed in IEC 62279 were classified according to the risk-management based categorisation scheme (Section 4.2.1). By referring to the information in the QAT Framework, we analysed the aims and descriptions of each candidate QAT to classify its role in managing PQRs. For each QAT, the descriptions were compared with the definitions and features of the risk-management based categories [3]. Some keywords/terms were used to identify the most relevant category for each QAT:

- PQR Assessment

  PQR Identification: e.g. identify, determine, find, discover cause/source, identify risk event (e.g. hazards (safety), critical use cases (performance)

  PQR Analysis: e.g. analyse consequences, analyse probability, analyse level of risk or hazards

  PQR Evaluation: e.g. evaluate/prioritise risks/potential problems, hazards (safety), critical use cases (performance)

- PQR Treatment:

  Runtime treatment: e.g. detect PQR event, protect, target, limit consequences, recovery routines, diagnose PQR event

  Design time treatment: Avoid/prevent safety or performance PQR event, remove/eliminate source of PQR

- PQR Monitoring: e.g. verify, validate, testing, monitoring, proof of correctness, proof of safety, proof of adequacy

When we analysed the candidate QATs, we found that many candidates had been classified under PQR treatment and PQR monitoring categories. The risk-management based categorisation scheme was modified significantly in order to apply it to the case study. Candidate QATs were clustered into smaller groups that have similar functions in managing PQRs. This helps to avoid the excessive cost of exhaustive pairwise comparisons at stage 3. Fig. 5 illustrates the QATs which fell under the PQR identification, PQR analysis and PQR evaluation categories. Fig. 6 and Fig. 7 illustrate clusters for each category. The definition of each cluster is described as follows:

1. PQR Treatment
   - Suitable programming languages: The programming languages chosen that can produce easily verifiable code with a minimum of effort and facilitate program development, verification and maintenance.
   - PQR treatment at design time: QATs which prevent PQR events or eliminate the source of PQRs at design time.
   - PQR treatment at run time: QATs which detect and diagnose PQR events, protect and limit consequences or recover systems to correct functional operation in the presence of PQRs at runtime.
2. PQR Monitoring
   - Independent verification and validation (V&V): Phase independent QATs which can be applied to monitor safety PQRs across the lifecycle.
   - Static analysis: QATs which monitor the PQRs by examining the properties of software systems without execution.
   - Dynamic analysis: QATs which monitor the PQRs by examining particular aspects (the code or a model of the code) of software systems under execution.

Table 4: Overview of the characteristics of each project

Characteristics	Project 1	Project 2	Project 3
Process Model	V-Model	V-Model	Iterative V-Model
Programming Languages Used	Ada 95, Delphi	C++, Java, VB, Pearl	C++
Team Size	4-30	20	8
Highest SWSIL	3 (High)	2 (Medium)	2 (Medium)

Candidate QATs in the same category or cluster were then compared according to process integration and cost/benefit perspectives to choose a combination of cost-effective QATs to assess and control PQRs across the software development lifecycle.

5.4.3. Stage 2: Compare Candidate QATs from Process Integration Perspective

Process information about QATs was used to integrate candidate QATs into software development lifecycle phases in which they could be used. The software development lifecycle model used by the three case projects was the V-Model. According to IEC 62279, Structured Methodology and Impact Analysis are highly recommended to be applied in Software Requirements Analysis and Software Maintenance phases. However, other QATs under the same clusters are not recommended to be used in these two phases. As a result, these two QATs were included separately from their clusters in Table 5.

5.4.4. Stage 3: Compare and Evaluate Candidate QATs from Cost/Benefit Perspective Using AHP

AHP [8] was used to evaluate the ranking of each candidate QAT in the same risk-management based category or cluster under cost/benefit selection criteria. The following five steps were performed to evaluate candidate QATs:

Step 1: Define the evaluation criteria used to select candidate QATs.

Selection criteria were defined for each project based on the questionnaire results (priority of the reasons for selecting and not selecting the QATs). The selection criteria for each project are as follows:

- Quality impact: Impact of QATs in managing PQRs (Projects 1, 2 and 3).
- Cost of application: The effort, time and resource required (Projects 1, 2 and 3).
- Expertise/Experience: The knowledge, experience or training required or available (Projects 1, 2 and 3).
- Ease of use: The complexity of applying the QAT (Projects 1, 2 and 3).
- Regulatory requirements: Recommendations by regulator/standard (Projects 1 and 3).
- Contractual requirements: Requirements specified by contract (Projects 1, 2 and 3).

Step 2: Weighting selection criteria using direct ratings.

In this case study, direct ratings were used to weight the selection criteria instead of pairwise comparisons. The relative importance of one criterion over another was expressed by comparing priority of selection criteria based on the questionnaire results. Reasons for selecting and not selecting the QATs for each project were collected from the questionnaire. For each criterion, the reasons for selecting and not selecting QATs were calculated based on the project responses in questionnaire for all the QATs under the same category. The ratio of one criterion over another was used as a relative weight to compare and evaluate candidate QATs in the same category. The main perceived selection criteria for each project were identified.

- PQR Assessment

  Project 1 - Contractual requirement: Expertise: Ease of use: Quality impact: Cost of application: Regulatory requirement = 14: 11: 7: 5: 5: 3 = 0.31: 0.24: 0.16: 0.11: 0.11: 0.07

  Project 2 - Cost of application: Expertise: Quality impact: Contractual requirement: Ease of use = 16: 14: 13: 9: 6 = 0.28: 0.24: 0.22: 0.16: 0.10

  Project 3 - Quality Impact: Cost of application: Expertise: Ease of use: Regulatory requirement = 13: 4: 4: 3: 1 = 0.52: 0.16: 0.16: 0.12: 0.04

- PQR Treatment:

  Project 1 - Cost of application: Contractual requirement: Expertise: Ease of use: Quality impact: Regulatory requirement = 23: 23: 23: 22: 14: 14 = 0.19: 0.19: 0.19: 0.18: 0.12: 0.12

  Project 2- Cost of application: Quality impact: Expertise: Contractual requirement: Ease of use = 33: 30: 21: 20: 16 = 0.275: 0.25: 0.175: 0.167: 0.133

  Project 3 - Expertise: Ease of use: Cost of application: Quality impact: Regulatory requirement = 18: 14: 12: 10: 1 = 0.33: 0.25: 0.22: 0.18: 0.02

- PQR Monitoring:

  Project 1 - Contractual requirement: Expertise: Cost of application: Quality impact: Regulatory requirement: Ease of use = 25: 21: 17: 13: 12: 11 = 0.25: 0.21: 0.17: 0.13: 0.12: 0.11

Table 5: Matching QAT clusters into the appropriate lifecycle phase(s) based on process information about QATs that summarised from literature and IEC62279

Phase	QAT Category		
	PQR Assessment	PQR Treatment	PQR Monitoring
System Development	Safety PQR Identification, Initial Safety PQR Analysis		Static Analysis, Independent V&V
Software Requirements Specification	Safety PQR Identification, Safety PQR Analysis, Safety PQR Evaluation	Structured Methodology	Static Analysis, Dynamic Analysis, Independent V&V
Software Architecture and Design	Safety PQR Analysis, Safety PQR Evaluation	PQR Treatment at runtime, PQR Treatment at design time	Static Analysis, Independent V&V
Software Module Design	Safety PQR Analysis, Safety PQR Evaluation	PQR Treatment at runtime, PQR Treatment at design time	Static Analysis, Independent V&V
Implementation	Safety PQR Analysis, Safety PQR Evaluation	PQR Treatment at runtime, PQR Treatment at design time, Suitable Programming Languages	Static Analysis, Dynamic Analysis, Independent V&V
Software Module Testing			Static Analysis, Dynamic Analysis, Independent V&V
Software Integration			Static Analysis, Dynamic Analysis, Independent V&V
Software/Hardware Integration			Static Analysis, Dynamic Analysis, Independent V&V
Software Validation			Static Analysis, Independent V&V
Software Assessment	Safety PQR Analysis Safety PQR Evaluation		Static Analysis, Independent V&V
Operations		PQR Treatment at runtime	Independent V&V
Maintenance	Impact Analysis		Independent V&V

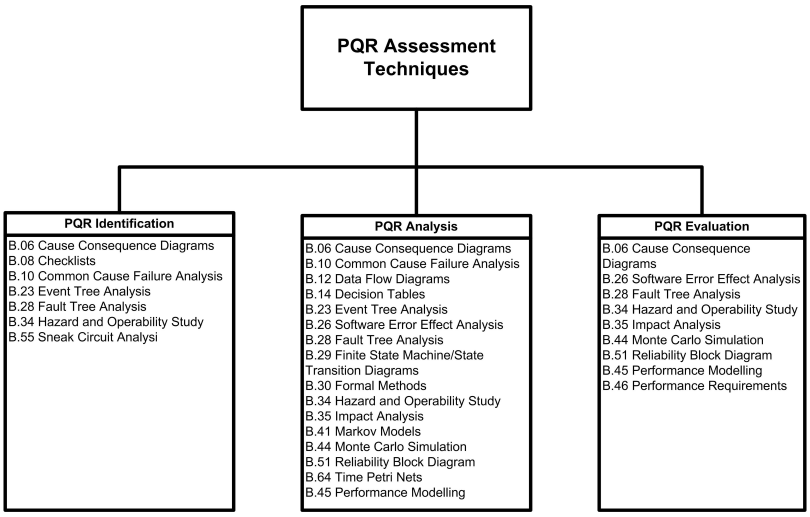

Figure 5: Candidate QATs for safety PQR assessment techniques

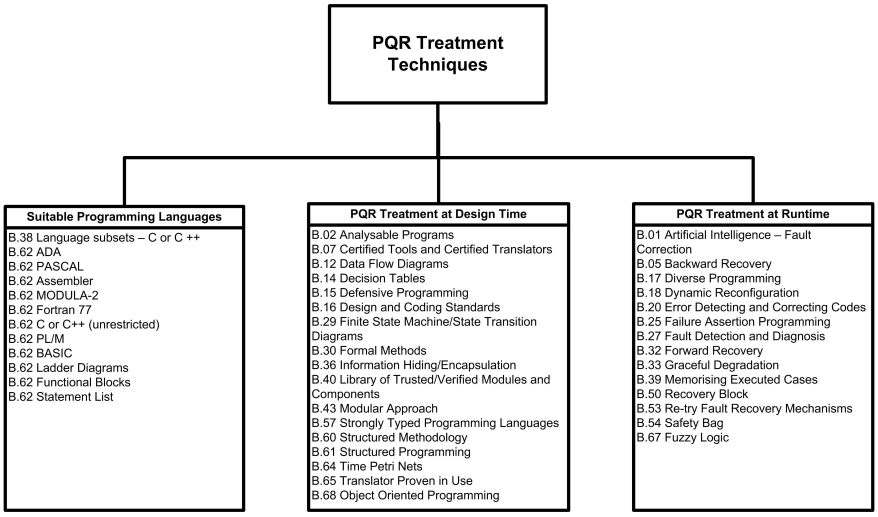

Figure 6: Candidate clusters for safety PQR treatment techniques

Project 2 - Cost of application: Expertise: Quality impact: Contractual requirement: Ease of use = 26: 19:17: 17: 8 = 0.30: 0.22: 0.19:0.19: 0.10

Project 3 - Quality impact: Expertise: Cost of application: Ease of use: Regulatory requirement = 15: 11: 8: 8: 1 = 0.35: 0.26: 0.19: 0.19: 0.02

Step 3: Determine relative ranking of each candidate QAT over another under each criterion using pairwise comparisons.

The candidate QATs in each cluster were evaluated in terms of how well they satisfy each selection criterion. For each criterion, a 9-point weighting scale [8] was used to weight the preference of each candidate over another in pairs. The information captured by the QAT Framework for each candidate helped to make pairwise comparisons. In this case study, pairwise comparisons were also made according to the standard recommendations, and project information analysed from questionnaire for the following criteria: team expertise, contractual requirement and quality impact. For each pair of candidate QATs, preference was determined based on the following QAT, Project and Standard (IEC 62279) information:

- Quality impact: (1) QAT Framework - aims, description, benefits and limitations of the QATs. (2) Project - industry experts' opinions for each project as indicated in the questionnaire.
- Cost of application: (1) QAT Framework - cost of application information captured by the QAT Framework; (2) Project - industry experts' opinions as indicated in the questionnaire.
- Expertise/Experience: (1) QAT Framework - the level of expertise or training captured by the QAT Framework; (2) Project - rankings by industry experts using a five-level scale (Not at all, Low, Medium, High, Very High).

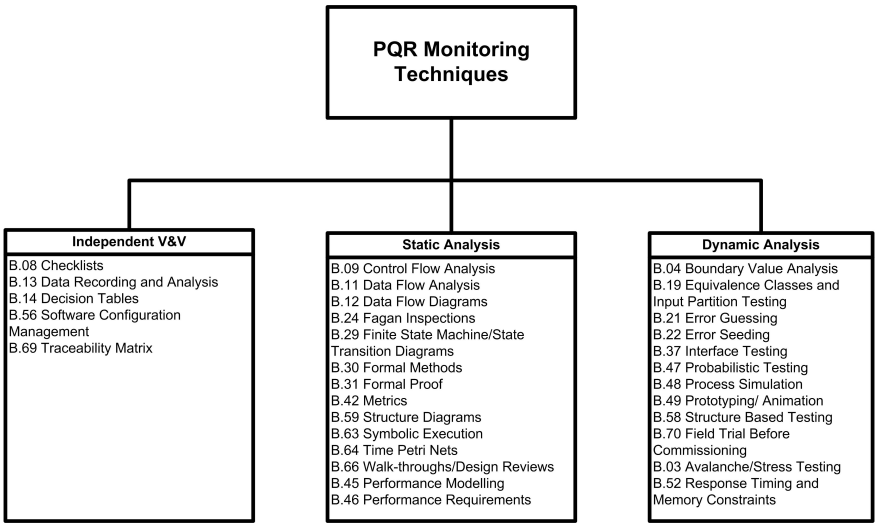

Figure 7: Candidate clusters for safety PQR monitoring techniques

- Ease of use: (1) QAT Framework - the complexity of applying the QATs was evaluated according to benefits and limitations captured by the QAT Framework; (2) Project - industry experts' opinions as indicated in the questionnaire.

- Regulatory requirements: (1) Standard - Recommendations (M: Mandatory; HR: Highly recommended; R: Recommended; –: No recommendation for or against being used; NR: Positively not recommended) provided by IEC 62279 to rank the appropriateness of the QATs for different SWSIL.

- Contractual requirements: (1) Project - each industry expert represented one project and reported whether QATs were explicitly required in the project's contract.

To illustrate the pairwise comparison between the candidate QATs, consider the comparison of Fault Tree Analysis (FTA) and Common Cause Failure Analysis (CCFA) on the expertise/experience criterion. The development team's capability for Project 1 was ranked "Very High" for FTA and "Low" for CCFA by the industry expert. As a result, FTA was considered strongly more important over the CCFA in terms of team expertise/experience.

Step 4: Compute the overall value score for each alternative with all criteria considered.

A pairwise comparison matrix was constructed for all candidate QATs in a cluster. Once the pairwise comparison matrix was formed, weights of candidates are calculated by solving for the eigenvector of the pairwise comparison matrix. The weights and utilities are calculated and transformed into candidates ranking (see Table 6).

Step 5: Perform sensitivity analysis.

Sensitivity analysis provides information on how the candidates ranking behaves in response to changes in priorities. For example, Fig. 8 illustrates a sensitivity analysis diagram of PQR identification QATs (Project 1) for criterion, Expertise. The current weight for this criterion is 0.24 and the current overall value score for FTA, Checklists, HAZOP, ETA, SCA (Sneak Circuit Analysis), CCFA (Common Cause Failure Analysis) and CCD (Cause Consequence Diagrams) are 30.58%, 22.04%, 11.58%, 10.26%, 9.90%, 8.57% and 7.07%. The diagram illustrates how the candidates ranking changed in response to the changes in weight priorities for the Expertise criterion.

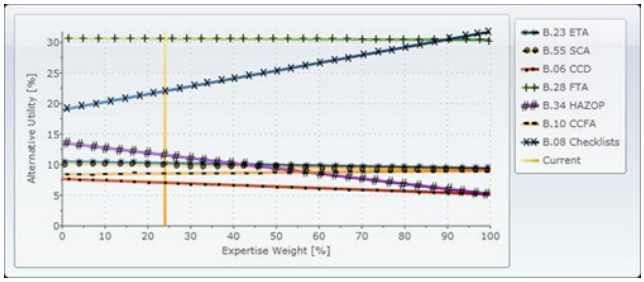

Figure 8: Sensitivity analysis diagram for criterion, Expertise (PQR Identification Techniques - Project 1)

6. Results

6.1. Candidates Ranking (Analysed using AHP)

Charts and matrices of candidates ranking for each project were produced using the analysis tool. An example of the candidates' ranking matrix and chart for PQR identification techniques of Project 1 are showed in Table 6 and Fig. 9.

6.2. Comparison

A combination of PQR assessment, treatment and monitoring QATs were selected based on the candidates ranking analysed by AHP [8]. When making the QAT selection

Table 6: Candidates ranking matrix for PQR identification techniques (Project 1)

Candidate	Total	Regulator	Contract	Expertise	Quality Impact	Ease of Use	Cost
B.28 FTA	30.58%	1.40%	14.09%	7.30%	5.00%	1.65%	1.13%
B.08 Checklists	22.04%	1.40%	2.82%	7.62%	1.00%	5.34%	3.87%
B.34 HAZOP	11.58%	0.47%	2.82%	1.24%	1.00%	3.52%	2.53%
B.23 ETA	10.26%	1.40%	2.82%	2.26%	1.00%	1.65%	1.13%
B.55 SCA	9.90%	0.47%	2.82%	2.17%	1.00%	2.31%	1.13%
B.10 CCFA	8.57%	1.40%	2.82%	2.17%	1.00%	0.63%	0.55%
B.06 CCD	7.07%	0.47%	2.82%	1.24%	1.00%	0.89%	0.65%

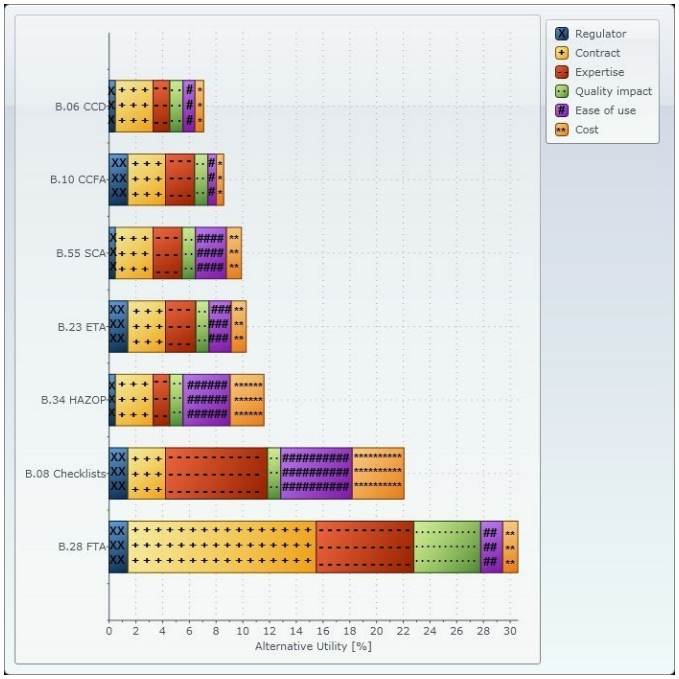

Figure 9: Candidates ranking for PQR identification techniques (Project 1)

decision for each QAT cluster, the candidate QATs with the highest aggregated utility were chosen. Other candidate QATs which also showed high aggregated utility were shortlisted for final consideration. The quality impact, regulatory requirement and contractual requirement of the shortlisted candidate QATs were analysed to decide whether the QATs should be included. For example, if a shortlisted QAT has significant quality impact in managing PQRs, is highly recommended by the standard or is required by the contract, this QAT will be selected. Fig. 10 shows a comparison of the QATs chosen using the QAT Selection Method and the actual QATs used in Project 1. Overall, the selection is broadly comparable with the actual QATs used in these projects. The differences will be discussed in the following sections for each project.

The actual lifecycle phase(s) that apply to the selected QATs were analysed. Most of the QATs were applied in the phases suggested by research literature and IEC 62279. However, there are some differences between the lifecycle phases recommended by the literature and the phases used in practice. In project 1, PQR treatment techniques were applied throughout the lifecycle phases. These QATs were used to generate test cases to verify and validate the product quality at later stages. Project 2 applied some PQR treatment techniques in early phases (system development and software requirements specification) based on the project needs. This project also applied Event Tree Analysis (ETA) to analyse safety PQRs during the Maintenance phase. Besides applying Design and Coding Standards (D&CS) during implementation, Project 3 also used them to generate test cases for module testing. According to IEC62279, the Impact Analysis technique was only recommended for Maintenance phase. In practice, development teams have applied this QAT across the software development lifecycle to analyse and evaluate PQRs, due to the effect that a change or an enhancement to a software system will have to other modules in that software system as well as to other systems.

6.2.1. Project 1

Although Checklists have different quality impact when applied in different phases, in this project they were only used for code review and for completion of code check-in. The impact of this QAT is through monitoring PQRs during the design, implementation and module testing phases. Checklists were not applied to identify PQRs. Although the development team has adequate expertise to apply Finite State Machine/State Transition Diagrams (FSM), this QAT was not selected because the legacy system designs are not suitable for using the QAT. Assembler was used in the actual project but was not selected by the Selection Method because the development team has low expertise with it. Nevertheless, in this project, assembler was required to implement certain hardware specific constructs, e.g. "test-and-set" operations used as the basis for implementing semaphores in shared memory. Another PQR monitoring technique called Automated Testing and Log Analysis was not chosen by the Selection Method because this QAT is not listed in IEC 62279. This QAT was used in the actual project.

- 93.5% of the techniques recommended by the selection method were actually used in the project
- 2.2% of the techniques recommended by the selection method were not used in the project

Category	Candidate Cluster	Selected QATs	Used in the Project?	Not selected but used in the project
PQR Assessment	PQR Identification	B.08 Checklists	No	
		B.23 FTA	Yes	
	PQR Analysis	B.23 FTA	Yes	
		B.35 IA	Yes	
		B.12 DFD	Yes	
		B.14 DT	Yes	
		B.44 MCS	Yes	
	PQR Evaluation	B.35 IA	Yes	
		B.23 FTA	Yes	
		B.46 PR	Yes	
		B.44 MCS	Yes	
PQR Treatment	Suitable Programming Languages	B.62 PASCAL	Yes	B.62 Assembler
		B.62 ADA	Yes	
	PQR Treatment at Design Time	B.16 D&CS	Yes	
		B.57 STP	Yes	
		B.07 CT&T	Yes	
		B.65 TPU	Yes	
		B.43 MA	Yes	
		B.12 DFD	Yes	
		B.15 DetP	Yes	
		B.68 OOP	Yes	
		B.61 SP	Yes	
		B.36 IH/E	Yes	
		B.02 AP	Yes	
		B.14 DT	Yes	
		B.40 LTM&C	Yes	
	PQR Treatment at Runtime	B.27 FD&D	Yes	
		B.25 FAP	Yes	
		B.53 RFRM	Yes	
		B.20 ED&CC	Yes	
PQR Monitoring	Static Analysis	B.11 DFA	Yes	
		B.42 Metrics	Yes	
		B.12 DFD	Yes	
		B.66 WT/DR	Yes	
		B.09 CFA	Yes	
		B.46 PR	Yes	
	Dynamic Analysis	B.48 PS	Yes	Automated Testing & Log Analysis (not listed in IEC62279)
		B.70 FTBC	Yes	
		B.04 BVA	Yes	
	Independent Verification and Validation	B.13 DR&A	Yes	
		B.56 SCM	Yes	
		B.08 Checklists	Yes	
		B.69 TM	Yes	
		B.14 DT	Yes	

Figure 10: Comparison of the QATs chosen by the Selection Method and the actual QATs used in Project 1

- 4.3% of techniques not recommended by the selection method were actually used in the project

6.2.2. Project 2

In project 2, there were a few QATs (Forward Recovery (FR), Dynamic Reconfiguration (DR) and BASIC) not recommended by the standard but nonetheless chose by the Selection Method for this project. FR was selected because it has a significant impact on quality objectives the cost of application is feasible, and the development team had adequate expertise/experience to apply this QAT. DR also has a significant impact on quality objectives and this QAT was required by contract. BASIC was selected by the Selection Method and also the development team but it has limited use in Project 2. The PQR monitoring technique, Metrics was used in the actual project but was not selected by the Selection Method because the development team had low expertise to perform this QAT. The development team had applied Functional Failure Analysis to identify and analyse PQRs and Hot Standby/Failover to treat the PQRs at runtime. These two QATs were not chosen by the method because they are not listed in the standard.

- 94.3% of the techniques recommended by the selection method were actually used in the project
- 0% of the techniques recommended by the selection method were not used in the project
- 5.7% of the techniques not recommended by the selection method were actually used in the project

6.2.3. Project 3

HAZOP, and Library of Trusted/Verified Modules and Components (LTM&C) were selected by the Selection Method but were not used in the actual project because they are covered by other PQR assessment and treatment techniques which have similar quality impact but are more cost-effective. Although the development team has adequate expertise to use Data Flow Diagrams (DFD), it was not selected because the role of this QAT in PQR management is covered by Sequence Diagrams. Finite State Machine/State Transition Diagrams (FSM) was selected but had very limited use and was only applied where there was a clear benefit.

- 94.9% of the techniques recommended by the selection method were actually used in the project
- 5.1% of the techniques recommended by the selection method were not used in the project
- 0% of the techniques not recommended by the selection method were actually used in the project

7. Discussion

This section discusses the effectiveness of decision making using the QAT Selection Method, uncertainty in decision making given changing priorities, and the applicability of the QAT Selection Method. The limitations of the study, and threats to construct, internal and external validity are also discussed.

7.1. Effectiveness of the decision-making process

The development teams originally selected their projects' QATs in an ad hoc manner based on their expertise and experience. In the case study, a set of QATs were selected for each project more systematically using the QAT Selection Method based on the three perspectives: PQR management, process integration and cost/benefit. Only a few QATs chosen by the QAT Selection Method were different from the actual selection. The risk-management based categorisation proved to be useful to compare and cluster the candidate QATs according to their functionality in managing PQRs. Additionally, categories or clusters of QATs were well integrated into software development lifecycle phases. Decision making

was supported by the rankings based on the cost/benefits perspective, to help select and shortlist QATs.

The QAT Selection Method identified the relative importance and position of QATs in the overall project context using AHP [8]. Three different combinations of QATs were selected based on the project needs. Project 1 and Project 2 developed real time systems. Also, the highest SWSIL of Project 1 is 3 (high). On the other hand, Project 3 developed offline design tools. Slow performance will not lead to safety issues. Response Timing and Memory Constraints (RT&MC) was the only performance related QAT that was selected by the QAT Selection Method for Project 3. Project 1 and 2 applied more QATs to ensure the safety of the software systems.

For these three projects, QATs such as Formal Methods, Fault Correction and Formal Proof were not selected because of the high level of expertise required to apply these QATs. The development teams were not sufficiently familiar with these QATs. It was too expensive and complex to apply them in the projects although these QATs were recommended by the standard. Other QATs which were more cost effective and had a similar quality impact were selected instead.

7.2. Uncertainty in Decision Making

Changes to the priorities in the decision criteria will affect the selection of QATs. The candidates' rankings (e.g. Fig. 9) show the sensitivity of the priority of these criteria. Tradeoffs analysis were made between conflicting selection criteria, with and without considering the priority weights. Tradeoffs highlight the most important criteria when the weights are prioritised. On the other hand, when the tradeoffs were analysed for candidate QATs without considering the priority weights, some of the candidates ranking were changed.

If there are new quality requirements or new PQRs introduced to the system, other candidates may have greater quality impact compared to the current ones. The initial choice of rankings may not be the best later on. This might be caused by team expectations for criteria change or the uncertainty of the initial decision. Development teams may only be aware of the actual impact of QATs after applying them. The effectiveness of applying selected QATs needs to be monitored. The lessons learned will be useful for future selection.

Development teams may have other options for using QATs to manage PQRs that are not listed in the standard. The standard IEC 62279 appears to reflect best practice of a few years ago rather than current best practice. Some commonly used or new QATs are not included in the standard, for example failure modes and effects analysis and the Java programming language. Also, some older QATs are rarely used in newly developed systems, and it can be now difficult to purchase a compiler/tool to support old QATs for a modern machine. There may be new standards or guidelines available in future which recommend new candidate QATs. The new options need to be considered for new projects.

7.3. Applicability

The case study of these three projects shows that it is feasible to apply the QAT Selection Method. QATs were selected to manage PQRs across lifecycle for each project. The QAT Selection method is a more systematic approach to choose appropriate QATs based on multiple criteria. The QAT information captured by the QAT Framework [47] was useful to aid in tradeoff analysis between QATs using pairwise comparisons. Information and categorisation of these QATs were recorded in catalogues which can be reused to select QATs for new projects which apply the same standard. Quantitative evaluation of candidate QATs using AHP [8] provided guidance for development teams to make decisions and explicit justifications for the selected QATs.

7.4. Limitations

Some weaknesses of the proposed approaches still remain and may limit their use in the following aspects:

- More time and effort are required to capture QAT information when the number of candidates is large. However, the information collected in the QAT Framework [47] can be reused to aid future selection.
- The work has not been validated through use on new projects. The validation has to date been post-hoc, by comparison with completed industry projects.
- To overcome the scalability issues of AHP [8], QATs have been divided into smaller clusters. However, the relationship between QATs in different clusters is not yet fully justified. For example, object oriented programming, defensive programming, structured programming, and programming languages were grouped into two different clusters (PQR Treatment at Design Time and Suitable Programming Languages) but nonetheless they are closely related.
- Since the candidate QATs are compared based on three perspectives, more time is required when the number of candidates is large. Some constraints may need to be added to shortlist the number of candidates (e.g. only to consider QATs for which the team has a sufficient level of expertise or only to consider QATs that are recommended for the SWSIL).
- The method only supports selection of QATs for an individual quality attribute. The tradeoffs among multiple quality attributes in QAT selection is not yet in scope of this QAT Selection Method.
- The method lacks mechanisms to select a balanced combination of inspection, modelling and testing QATs, to avoid over-reliance on human based QATs.

7.5. Threats to validity

This section discusses threats to construct, internal and external validity for the case study. Countermeasures against threats to validity were taken and are described.

Construct validity threats concern whether the operational measures that are studied really represent what the researchers have in mind and what is investigated according to the research questions [55]. For example, the constructs discussed in the interview may be misinterpreted by the interviewed persons. To limit these threats, a presentation was given before the participants filled in the questionnaire, to reduce the likelihood of misinterpreting the questions. Additionally, a questionnaire guide and training documents were provided to assist the participants in understanding the proposed work and questionnaire. The researchers were present when the participants answered the questionnaire, so participants could ask questions if they were not clear about any issue. Threats to construct validity are also partly countered by using established constructs from risk management theory and the literature.

Internal validity threats concern the causal relations that may affect the outcome of this study [55]. To improve the precision of this case study, data and method triangulation was achieved in different ways. A combination of qualitative and quantitative methods was applied to collect and analyse data. The questionnaire and interview were two main data collection methods. The follow-up reports were sent to participants to clarify the issues identified through questionnaire analysis. Two researchers conducted the interview together, the interview was audio recorded, and later transcribed. Additionally, two researchers had classified a list of QATs independently based on risk-management based categorisation scheme. The categories assigned to each QAT were compared and finalised in a joint meeting. Participants were not aware of the case study objectives even if it was clear to them our intention to validate the QAT Selection method.

External validity threats concern the generalisation of the findings [55]. Although only one case study was conducted to validate the QAT Selection Method, three projects were selected to replicate and cross-validate the results of the evaluation. Only one organisation was selected to conduct the case study but literature-based evaluation was conducted in an earlier study to validate the feasibility of the QAT Selection Method. Although this case study only validated the proposed work using safety quality, the QAT Selection Method is intended to be to be able to be applied to other quality attributes. The earlier literature-based study [3] fit other quality QATs to the framework.

8. Conclusion

This research aims to better understand how to select techniques to manage product quality risks throughout software development lifecycle. Risk management theory was applied to categorise and compare the QATs for arbitrary quality attributes. A QAT Selection Method was proposed to help development teams compare and choose QATs based on three perspectives: product-quality-risk management, process integration and cost/benefit. For the PQR management perspective, risk management theory was used to compare the candidate QATs based on their intended used in addressing PQRs. For the process integration perspective, QATs are incorporated into software lifecycle phases that can be applied to help development teams choose QATs with different impact in managing PQRs, across the lifecycle. For the cost/benefit perspective, Analytic Hierarchy Process (AHP) [8] was used to compare and evaluate the candidate QATs.

The QAT Selection Method was evaluated in a case study. For the case study, three previously-completed projects from an organisation that develop safety-related systems were selected. The case study was run with three industry experts, each representing a project. In the case study, the QAT Selection Method was applied to retroactively choose QATs for the projects. The applicability of the QAT Selection Method was supported because the QATs selected by the method were broadly comparable with the QATs that had actually been selected. The QAT Selection Method also provides a more systematic approach to choose QATs than selection of QATs by experts and can provide explicit justification for the selection.

The following sections discuss the implications of this work for research and practice, identify the limitations and propose future research.

8.1. Implications for Research

Software quality is determined during the software development process, but prior research on software process modelling and tailoring had not provided an explanation of how different software processes target specific software qualities. QATs are widely used in practice and play an important role in targeting specific product qualities, but prior research had not explained how QATs fit into software process models. This research provides an improved understanding about the relationship between QATs, software process and product quality, through a risk management perspective.

- The research provides a new way of thinking about how specific product quality attributes can be targeted, by identifying, analysing, evaluating, controlling, and monitoring potential risks to product quality that can arise during software development.
- The classification of QATs in a product-quality-risk management lifecycle provides a new basis for understanding how QATs fit into the traditional software development lifecycle process.
- The QAT Selection Method provide a basis to support future research in process tailoring, where a key

tailoring objective is product quality, not just project or organisational context.

8.2. Implications for Practice

The QAT information captured and presented using the QAT Framework [47] is motivated by the intention to help practitioners to understand how QATs help to achieve the product quality requirements and how selected QATs can be incorporated into their process models. This research shows how QATs can be selected based on their impact in managing PQRs. The QAT Selection Method offers a more systematic approach for software development teams to compare candidate QATs and choose QATs that best fit their product quality requirements and project context. The QAT Selection Method does not replace experts in decision-making for QAT selection. It rather helps them to:

- Make informed decisions for the selected QATs through quantitative evaluation of candidate QATs using AHP [8]. The method also provides an explicit way to rationalise and justify these decisions.
- Examine and evaluate all candidates and explore the tradeoffs between available candidate QATs based on risk-management, process integration and cost/benefit perspectives.
- Determine a combination of QATs that covers the PQR management lifecycle throughout the software development lifecycle.

8.3. Future Work

There are many opportunities to extend this research and overcome some of its limitations. The first area of future work is to widen and deepen the empirical evaluation of the QAT Selection Method. This work is intended to be able to cover any quality attribute, and so the QAT Selection Method should be applied and evaluated for other quality attributes.

The work could be extended by the development of related methods and tools. Further work is required to develop a comprehensive process tailoring method to support integration of QATs into process models. The current catalogue of QATs is stored in Microsoft word documents. A repository tool could be developed to facilitate the maintenance and dissemination of QAT information.

Currently, the work only supports selection of QATs for individual quality attributes. Further work would be required to be able to simultaneously select and integrate QATs into software processes for multiple quality attributes. AHP [8] is used to evaluate the tradeoffs among QATs against cost/benefit criteria. The use of candidate multi-criteria decision analysis methods could be explored to improve the efficiency in selection.

The process of managing PQRs is similar to the generic risk management process. Future work would be required to better integrate the process of managing PQRs to target specific product quality requirements throughout software development lifecycle process. Also, the QAT Selection Method can be improved to support selection of a balanced set of human-based, formal methods and informal QATs across the lifecycle.

Acknowledgments

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] L. Zhu, R. Jeffery, M. Staples, M. Huo, T. T. Tran, Effects of Architecture and Technical Development Process on Micro-process, in: International Conference of Software Process (ICSP), 49–60, 2007.

[2] L. Zhu, T. T. Tran, M. Staples, R. Jeffery, Technical Software Development Process in the XML Domain, in: International Conference of Software Process (ICSP), 246–255, 2009.

[3] Y. K. Chiam, L. Zhu, M. Staples, Quality Attribute Techniques Framework, in: 16th European Conference (EuroSPI), 173–184, 2009.

[4] V. R. Basili, H. D. Rombach, Tailoring the software process to project goals and environments, in: Proceedings of the 9th International Conference on Software Engineering, 345–357, 1987.

[5] J. Bowers, J. May, E. Melander, M. Baarman, A. Ayoob, Tailoring XP for Large System Mission Critical Software Development, in: Second XP Universe and First Agile Universe Conference, 100–111, 2002.

[6] O. Pedreira, M. Piattini, M. R. Luaces, N. R. Brisaboa, A Systematic Review of Software Process Tailoring, ACM SIGSOFT Software Engineering Notes 32 (3) (2007) 1–6.

[7] X. Ferre, N. Juristo, A. M. Moreno, Framework for Integrating Usability Practices into the Software Process, in: International Conference on Product-Focused Software Development and Process Improvement (PROFES), 202–215, 2005.

[8] T. L. Saaty, The Analytical Hierarchy Process, McGraw-Hill, 1980.

[9] Y. K. Chiam, L. Zhu, M. Staples, Systematic Selection of Quality Attribute Techniques, in: International Conference on Product Focused Software (PROFES), 59–62, 2010.

[10] J. Holt, Current practice in software engineering: a survey, Computing & Control Engineering Journal 8 (4) (1997) 167–172.

[11] K. E. Emam, A. Birk, Validating the ISO/IEC 15504 Measure of Software Requirements Analysis Process Capability, IEEE Transactions on Software Engineering 26 (6) (2000) 541–566.

[12] C. Jones, Software Assessments, Benchmarks, and Best Practices, Addison-Wesley, 2000.

[13] R. L. Glass, Matching Methodology to Problem Domain, Communications Of The ACM 47 (5) (2004) 19–21.

[14] L. Jiang, A Framework for the Requirements Engineering Process Development, Ph.D. thesis, Department of Electrical and Computer Engineering, University of Calgary, Canada, 2005.

[15] D. Damian, J. Chisan, An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management, IEEE Transactions on Software Engineering 32 (7) (2006) 433–453.

[16] IEC 61508-3, Functional safety or electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements, 1998.

[17] IEC 62279, Railway Applications Communications, Signalling and Processing Systems Software for Railway Control and Protection Systems., 2002.
[18] Zurich Risk Engineering, Which Hazard Analysis? - A Selection Guide, 1998.
[19] W. Bridges, Selection of Hazard Evaluation Techniques, http://www.piii.com/_downloads/Selection of Hazard Evaluation Techniques.pdf, 2004.
[20] M. Lyons, Towards a framework to select techniques for error prediction: Supporting novice users in the healthcare sector, Applied Ergonomics 40 (3) (2009) 379–395.
[21] W. E. Perry, A Structured Approach to Systems Testing, Prentice-Hall, 1983.
[22] S. Vegas, V. Basili, A Characterisation Schema for Software Testing Techniques, Empirical Software Engineering 10 (4) (2005) 437–466.
[23] L. Jiang, A. Eberlein, B. H. Far, Combining Requirements Engineering Techniques - Theory and Case Study, in: IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS), 105–112, 2005.
[24] M. Svahnberg, C. Wohlin, L. Lundberg, M. Mattsson, A Quality-Driven Decision-Support Method for Identifying Software Architecture Candidates, International Journal of Software Engineering and Knowledge Management 13 (5) (2003) 547–573.
[25] T. Al-Naeem, I. Gorton, M. A. Babar, F. Rabhi, B. Benatallah, A quality-driven systematic approach for architecting distributed software applications, in: International Conference on Software engineering (ICSE), 244–253, 2005.
[26] L. Zhu, A. Aurum, I. Gorton, R. Jeffery, Tradeoff and Sensitivity Analysis in Software Architecture Evaluation Using Analytic Hierarchy Process, Software Quality Journal 13 (4) (2005) 357–375.
[27] EWICS TC7 Software Sub-group, Techniques for Verification and Validation of Safety-related Software, Computers and Standards 4 (2) (1985) 101–112.
[28] M. A. Babar, L. Zhu, R. Jeffery, A Framework for Classifying and Comparing Software Architecture Evaluation Methods, in: Australian Software Engineering Conference (ASWEC), 309–318, 2004.
[29] M. A. Babar, B. Kitchenham, P. Mehashwari, Assessing the Value of Architectural Information Extracted from Patterns for Architecting, in: Proceedings of the Empirical Assessment in Software Engineering (EASE), 1–10, 2006.
[30] H. D. Rombach, Systematic Software Technology Transfer, in: International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions, 239–246, 1993.
[31] S. Vegas, Characterisation Schema for Selecting Software Testing Techniques, Ph.D. thesis, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain, 2002.
[32] D. Jewell, Performance Engineering and Management Method - A Holistic Approach to Performance Engineering, in: Performance Modeling and Engineering, 29–55, 2008.
[33] B. Boehm, Software Risk Management: Principles and Practices, IEEE Software 8 (1991) 32–41.
[34] B. Boehm, A Spiral Model of Software Development and Enhancement, IEEE Computer 21 (1998) 61–72.
[35] R. Charette, Software Engineering Risk Analysis and Management, McGraw-Hill, 1989.
[36] R. Charette, Large-Scale Project Management is Risk Management, IEEE Software 13 (1996) 110–117.
[37] Jones, P. L. and Jorgens, J. III and Taylor, A. R. Jr and Weber, M., Risk management in the design of medical device software systems, Biomedical Instrumentation & Technology 36 (2002) 237–266.
[38] F. M. Caffery, J. Burton, I. Richardson, Risk management capability model for the development of medical device software, Software Quality Control 18 (2010) 81–107.
[39] NASA Aeronautics and Space Administration, NASA Software Safety Guidebook, NASA-GB-8719.13, 2004.
[40] R. Keeney, H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Trade-Offs, Cambridge University Press, 1993.
[41] B. Roy, Multicriteria Methodology for Decision Aiding, Kluwer Academic Publishers, 1996.
[42] R. Keeney, Foundations for Making Smart Decisions, IIE Solutions 31 (5) (1999) 24–30.
[43] R. Fuller, C. Carlsson, Fuzzy multiple criteria decision making: Recent developments, Fuzzy Sets and Systems 78 (2) (1996) 139–153.
[44] J. Karlsson, K. Ryan, Cost-value approach for prioritizing requirements, IEEE Software 14 (5) (1997) 67–74.
[45] E. Hotman, Base Reference Analytical Hierarchy Process for Engineering Process Selection, in: R. Khosla, R. Howlett, L. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, vol. 3681 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 155–155, 2005.
[46] D. Opydo, MakeItRational, URL `http://makeitrational.com`, http://makeitrational.com.
[47] Y. K. Chiam, L. Zhu, M. Staples, Quality Attribute Techniques Framework, in: R. V. OConnor, N. Baddoo, J. Cuadrago Gallego, R. Rejas Muslera, K. Smolander, R. Messnarz (Eds.), Software Process Improvement, vol. 42 of *Communications in Computer and Information Science*, Springer Berlin Heidelberg, 173–184, 2009.
[48] N. G. Leveson, Safeware: System Safety and Computers, Addison Wesley, 1995.
[49] AS/NZS ISO/IEC 12207, Software Life Cycle Processes, 1997.
[50] Y. K. Chiam, Representation and Selection of Quality Attribute Techniques for Software Development Process, Ph.D. thesis, School of Computer Science and Engineering, University of New South Wales, Australia, 2011.
[51] N. Storey, Safety Critical Computer Systems, Addison Wesley, 1996.
[52] J. Borcsok, S. Schaefer, Software Development for Safety-related Systems, in: International Conference on Systems (ICONS), 38–42, 2007.
[53] S. Vegas, Identifying the Relevant Information for Software Testing Technique Selection, in: International Symposium on Empirical Software Engineering (ISESE), 39–48, 2004.
[54] R. K. Yin, Case Study Research: Design and Methods, Sage Publications, 4th edn., 2008.
[55] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering 14 (2009) 131–164.