Information and Software Technology 54 (2012) 651–662

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

# Software quality trade-offs: A systematic map

Sebastian Barney [a,b,*], Kai Petersen [a], Mikael Svahnberg [a], Aybüke Aurum [b], Hamish Barney [b]

[a] *Blekinge Institute of Technology, Sweden*
[b] *School of Information Systems, Technology and Management, University of New South Wales, Australia*

ARTICLE INFO

*Article history:*
Received 2 August 2011
Received in revised form 20 January 2012
Accepted 23 January 2012
Available online 2 February 2012

*Keywords:*
Software engineering
Software quality
Trade-off approaches

ABSTRACT

*Background:* Software quality is complex with over investment, under investment and the interplay between aspects often being overlooked as many researchers aim to advance individual aspects of software quality.
*Aim:* This paper aims to provide a consolidated overview the literature that addresses trade-offs between aspects of software product quality.
*Method:* A systematic literature map is employed to provide an overview of software quality trade-off literature in general. Specific analysis is also done of empirical literature addressing the topic.
*Results:* The results show a wide range of solution proposals being considered. However, there is insufficient empirical evidence to adequately evaluate and compare these proposals. Further a very large vocabulary has been found to describe software quality.
*Conclusion:* Greater empirical research is required to sufficiently evaluate and compare the wide range of solution proposals. This will allow researchers to focus on the proposals showing greater signs of success and better support industrial practitioners.

**Contents**

* Corresponding author at: School of Information Systems, Technology and Management, University of New South Wales, Australia. Tel.: +61 2 93857124.
*E-mail addresses:* s.barney@unsw.edu.au (S. Barney), kai.petersen@bth.se (K. Petersen), mikael.svahnberg@bth.se (M. Svahnberg), aybuke@unsw.edu.au (A. Aurum), hamish@student.unsw.edu.au (H. Barney).

0950-5849/$ - see front matter Crown Copyright © 2012 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2012.01.008

## 1. Introduction

Software quality is far more complex than acknowledged by much of the literature addressing the topic. The interplay between software quality aspects are often overlooked as many researchers aim to increase individual aspects of software quality. The problems associated with under investment are obvious, but over investment is also problematic [57,52]. Insufficient quality leads to a useless product. As the level of quality increases the product become useful, and some point the level of quality can give the product a competitive advantage. However, beyond a certain point the level of quality becomes excessive—requiring high levels of investment, but not providing any competitive advantage, or sufficient business value to offset the required cost.

Given both the importance and complexity involved in achieving the right balance of software quality there is value in consolidating the research addressing software quality trade-offs. Consolidating the knowledge will help practitioners access the approaches to software quality trade-off that are most suitable to their given context, and identify research gaps that academics can focus greater study. Thus, this paper aims to provide an overview of the approaches used in software quality trade-offs, and the level of empirical evidence currently provided.

A systematic map of the software quality trade-off literature has been developed to understand the state of the research addressing this area. The analysis was done in two phases, with *Phase 1* providing an overview of the literature addressing the topic in general and *Phase 2* provding an overview of the empirical literature addressing the topic.

The remainder of this paper is structured as follows. The background to the study is presented in Section 2. The systematic mapping process is detailed in Section 3. The results and discussion for *Phase 1* are presented in Section 4 with the *Phase 2* presented in Section 5. Conclusions drawn in Section 6.

## 2. Background

Quality is a complex and multifaceted concept [22]. This holds true for software.

The software development industry has traditionally defined software quality as ‘fit for purpose’ or ‘conforming to specification’ [26]. Most research on software quality focuses on improving and optimising individual aspects of software product quality (e.g. maintainability, security, or usability). However, achieving the highest level of quality against a model or set of measures does not ensure a sufficient level of quality will be achieved [57]—it is possible that the required level of quality is higher than that assumed by any given model or measure. Further, such sub-optimisation of software product improvements does not always make sense. Improving one aspect of software quality can cause improvements or degradation to other aspects of software quality [31,57].

There are numerous models of software quality to support the software development process [33]. Commonly cited models include McCall’s Quality Model, Boehm’s Quality Model, Dromey’s Quality model and ISO9126 [29]. Most quality models are presented in a hierarchy of attribute, sub-attributes and occasionally metrics. Further, it is recommended that any quality model should be tailored to any specific context in which it is used.

More recently software engineering research has emphasised the need to consider software quality from a range of different perspectives and take a value-based approach to software quality [9]. This means ensuring that the level of quality delivered is acceptable to all the stakeholders upon whom the success of the product depends.

The value of a shared understanding of software quality between key stakeholders is increasingly being recognised. Organisations with highly aligned individuals and groups have been found to significantly outperform those organisations that do not share these high levels of alignment [13]. However, achieving and maintaining alignment is an ongoing activity as priorities will naturally change over time for numerous reasons [12].

There are numerous approaches to reconciling conflicts between between aspects of software product quality. Some of the most common include expert judgement, the Non-Functional Requirements (NFR) Framework, Quality Function Deployment (QFD) and Theory W.

Expert judgement involves one or more experienced professionals using their experiences and knowledge to make a decision on an issue. The decisions are not necessarily supported by modelling or numerical assessment.

The NFR Framework uses diagrams to relate non-functional requirement goals with different decisions that can be made in the design and operation of a system that affect it positively or negatively, allowing trade-offs to be identified and made [15]. While this method makes the results of a choice to be made more explicit, it requires a set of common priorities to be identified to allow effective decisions to be made.

Quality function deployment (QFD) considers the priority of customer and technical requirements in achieving the goals of the system to help prioritise the requirements [24]. However, the other perspectives involved in the development of the software product are not considered.

Value-based software engineering (VBSE) recognises the problems created by conflicting perspectives in the software development process [8]. Central to resolving conflict in VBSE is Theory-W [9], which involves identifying the success-critical stakeholder, determining a mutually acceptable set of objectives, and then working to ensure these objectives are achieved.

### 2.1. Summary of previous reviews

At the time of writing the authors were only able to find one systematic review addressing the issue of software quality trade-off decisions and analysis—published by Berntsson Svensson et al. [6]. This publication addresses the more general topic of software quality management, but specifically aims to address issues of prioritisation between aspects of software quality. However, on the topic of software quality prioritisation this review was only able to identify three relevant articles. This weakness is caused by the limited number of keywords used to define the prioritisation concept within the search string.

The authors hope that by conducting a more focused and detailed study, the topic of software quality trade-off decisions and analysis can be better understood.

### 2.2. Research questions

The aims of this paper are twofold.

In *Phase 1* this paper aims to provide an overview of the recent literature addressing software quality trade-offs. This aim is represented by the first research question, which is further divided into sub-research questions:

- **RQ1.** What is the state of the art with respect to software quality trade-off literature?
  – **RQ1.1.** What are the common publication fora?
  – **RQ1.2.** Who are the key researchers in the field?
  – **RQ1.3.** Which research approaches are employed?
  – **RQ1.4.** During which development phase does literature suggest for software quality trade-offs to takes place?
  – **RQ1.5.** Which approaches are presented in the literature?

*Phase 2* of the research aims to provide a specific description of the empirical literature addressing software quality trade-offs. This aim is presented as the second research question and divided into sub-research questions:

- **RQ2.** What is the state of the art for empirical research addressing software quality trade-offs?
  – **RQ2.1.** Which approaches in the literature have been empirically studied?
  – **RQ2.2.** What are the aims of the empirical work?
  – **RQ2.3.** What level of academic rigour and industrial evidence is present in the empirical work?

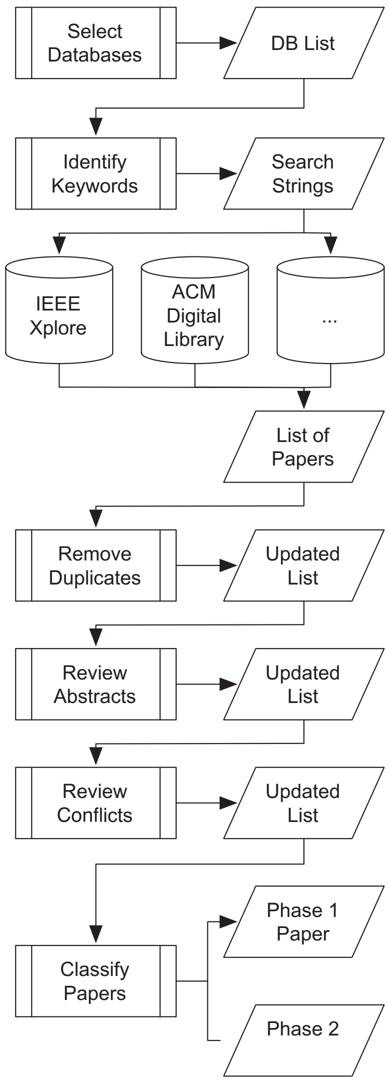

**Fig. 1.** Research process.

## 3. Method

The systematic mapping process is used to address the research questions posed in this paper. A systematic map can provide an overview of the literature addressing software quality trade-offs and can also be used to identify empirical publications addressing software quality trade-off literature. It can then be used to provide a detailed description of the identified publications, answering RQ2.

### 3.1. Research process

The methodology employed in this work is based on the systematic review guidelines presented by Kitchenham [32]. It has been further informed and shaped by the experiences of Dybå and Dingsøyr [20], Petersen et al. [47], Riaz et al. [53].

The following procedure was followed in conducting the systematic mapping:

1. First bibliographic databases were selected from which to search for publications. The rationale and choices are presented in Section 3.2.
2. Keywords describing the research area were identified, and search strings were created for the selected databases. This process is described in Section 3.2.
3. The search strings were then run against the databases, with the bibliographic information for all resultant publications extracted and saved.
4. The list of publications from each database were combined, with duplicate records for a publication removed.
5. The title and abstracts of each publication was then reviewed by two reviewers against a set of inclusion and exclusion criteria. This process is described in Section 3.3.
6. Publications that were marked for both inclusion and exclusion were then re-reviewed and selected for inclusion or exclusion, as described in Section 3.3.
7. The resultant list of publications were then classified for the systematic map. This process is described in Section 3.4.
8. Finally a data extraction was undertaken on the relevant publications. This process is described in Section 3.5.

A summary of this process is provided in Fig. 1.

### 3.2. Databases, keywords and search strings

After defining the research questions, the first step of conducting the systematic map and review was to identify the set of databases to be used to find publications. As this systematic review addresses software engineering, databases were selected for their coverage and use in this domain. The following set of databases were selected to identify relevant publications:

- ACM Digital Library
- Compendix/Inspec
- IEEE Xplore
- ISI Web of Science
- Scopus

While the authors wanted to include *SpringerLink*, this database was excluded due to its inability to handle complex queries.

The next step was to identify the set of keywords. These keywords are used to create search strings, which are run against the selected publication databases.

Keywords were identified in an iterative approach with several steps. First, relevant papers were identified, and appropriate

**Table 1**
Search string keywords.

| Category | Label | Keyword(s) |
|---|---|---|
| C1 | Software | Software |
| C2 | Engineering | Engineering, development, product, service, system |
| C3 | Product Quality | ISO9126, ISO 9126, ISO-9126, IEC9126, IEC 9126, IEC-9126, non-functional+(artefacts, artifacts, aspects, attributes, capabilities, characteristics, factors, features, properties, requirements, traits), all ISO 9126 quality aspect pairs |
| C4 | Trade-offs | Tradeoff, "trade off", "trade-off", priorit*, balance, collaborat*, compromis*, "cumulative voting", "analytic hierarchy process", AHP, vote, voting |

**Table 2**
Results from each database.

| Database | Publications |
|---|---|
| ACM Digital Library | 113 |
| Compendix/Inspec | 493 |
| IEEE Xplore | 965 |
| ISI Web of Science | 108 |
| Scopus | 1248 |
| Total | 2930 |

keywords were extracted. Then thesauri were used to identify additional keywords based on the extracted keywords. Search strings were then created and run against various databases to identify additional relevant publications. This process was repeated until there were no significant developments to the keywords. This process was conducted by the first author, and then reviewed by coauthors.

The keywords were classified into four groups—*software* (C1), *engineering* (C2), *product quality* (C3) and *trade-offs* (C4). These categories and the final list of keywords are presented in Table 1. To increase readability, all of the three word combinations starting *non-functional* or *non functional* are listed together without repeating the the first two words. Further, each pair of ISO 9126 aspects was inserted. The `AND` operator was used between the two items making each pair, while the `OR` operator was used between each pair.

The search string was created by putting the `OR` operator between all of the words or phases within each category, and putting the `AND` operator between each category. Thus the main search string was composed as:

```
C1 AND C2 AND C3 AND C4
```

where, for example, C2 was composed as:

```
(engineering OR development OR product OR service OR system)
```

The search queries were further limited to publications printed in 2005–2010. The authors considered this six-year period to cover 'recent' publication, and that any previously identified approach not revisited during this time unlikely to be still be considered relevant.

Applying the searches created from this process to the selected databases yielded 2930 publication, as presented in Table 2. Combining the results from each database and removing duplicates reduced this list to 2153 publications. However, further duplicates were found later in the process.

## 3.3. Inclusion and exclusion criteria

The 2153 publications identified by the search strings were then reviewed against a set of inclusion and exclusion criteria.

The inclusion and exclusion criteria were developed in a review of publications from 2009 by the first author. Articles were reviewed with rules being developed and refined to support the decision making process. These rules were then reviewed and agreed upon by the coauthors.

The final set of rules, applied to the review of all abstracts, is set out below:

- Included results must be in English.
- Included results must not come from an excluded source:
  - Books are excluded, however, book chapter from an edited volume may be included.
  - Forewards are excluded regardless of their source.
  - Editorials are excluded.
- Included results must address software development.
- Included results must address multiple aspects of software product quality (for example, see ISO9126 [29]).
  - Aspects of product quality not relating to a software product are out of scope in this study (e.g. the reliability of a car is out of scope, but the reliability of the software running a car is in scope).
  - Aspects of service quality that do not directly relate to software product quality are out of scope in this study (e.g. reliability can be considered an aspect of service quality and product quality, customer service does not).
  - Aspects of process quality that do not directly relate to software product quality are out of scope in this study (e.g. having a reliable process is different to having a reliable product).
- Included results must address trade-offs between at least two aspects of software product quality.
  - Results that allow direct comparisons between aspects of software product quality can be included in this study.
  - Results only seeking to improve one or more aspects of software product quality without a trade-off between them are excluded from this study.
  - Results that perform trade-offs between *a single* aspect of software product quality and some aspect non-software product quality aspects are excluded.
  - Results that perform trade-offs between *multiple* aspects of software product quality and an other aspect or aspects are included.

Each publication received two independent reviews against the inclusion and exclusion criteria. The first author reviewed all publications, while the other four authors each reviewed one quarter of the publications. In this process 140 publications were marked *in* scope, while 1866 publications were marked as *out* of scope and 147 publications had conflicting reviews.

Where the two reviews were in agreement a publication no further action was taken, with the paper being marked in or out of scope as appropriate. The 147 publications with conflicting reviews were given to the second and third authors to get a third and final assessment. The second author reviewed all 133 publications not previously reviewed by this author, marking 36 *in* scope.The third author reviewed the 14 publications with conflicting reviews that had previously been reviewed by the second author, marking 3 *in* scope. This resulted an additional 39 publications being marked

**Table 3**
Reasons for exclusion during mapping process.

| Reason | Number |
|---|---|
| Less than two quality aspects studied | 3 |
| Previously unidentified duplicates | 2 |
| Process quality, not software quality | 2 |
| Foreward (excluded source) | 2 |
| Publication not in English | 1 |

*in* scope, giving a grand total of 179 publications being marked *in* scope.

### 3.4. Data extraction for Phase 1

Having identified the papers in scope, the next step was to classify the publications for the systematic map—answering RQ1. For each publication the following data was collected from the abstract, unless otherwise stated:

- *Qualities*: The aspects of quality considered as part of the trade-off in the publication.
- *Trade-off approach*: The approach used to support software quality trade-offs. If this could not be determined from the abstract, the introduction and conclusion where read.
- *Development artefact*: The development artefact to which the trade-off was applied. The options considered were *process*, *requirements*, *architecture*, *code*, *test* and *runtime*.
- *Research approach*: The research approach used according to the classification system proposed by Wieringa et al. [61], as recommended in Petersen et al. [47]. The research approaches are *opinion papers*, *philosphical papers*, *experience reports*, *solution proposals*, *validation research* and *evaluation research*. Where this could not be confirmed, the introduction and conclusion were read.
- *Authors*: The authors who wrote the publication.
- *Venue*: The publishing journal, conference, workshop or book.
- *Year*: The year of publication.

Given this study assesses peer-reviewed academic publications, trust was placed in the ability of the authors to correctly classify their own work with the support of the reviewers. For example, if an author stated that a case study was conducted, this information is assumed correct and was not the subject of further validation.

An iterative classification process was used when collecting data on the *qualities considered* and *trade-off approach*. The qualities and approach studied in the first paper were documented in separate lists. Each subsequent publication was assessed to determine if it fitted within the list or, if required, a new item was added to the appropriate list.

During the classification process 10 publications were excluded for the reasons stated in Table 3. An additional publication was removed during the *Phase 2* analysis as it was found to apply approaches used in software industry to physical products. The final list of publications totalled 168.

### 3.5. Data extraction for Phase 2

Each publication classified by the systematic mapping process as empirical research was the subject of further study in *Phase 2*.

The full publication was reviewed, and the following information was collected:

- The aim of the doing the software quality trade-off.
- The approach supporting the software quality trade-off.
- A brief description of the process used to conduct the software quality trade-off.
- A brief description of the empirical evidence presented in the paper.
- A classification of the paper against the rigour and relevance criteria for empirical research in software engineering, as proposed by Ivarsson and Gorschek [30].

In total 49 publications were identified from *Phase 1* as within scope of *Phase 2*, however, six of these papers were excluded upon more detailed inspection:

- The full-text of one publication could not be obtained online or after making contact with the authors. Thus, this publication was excluded as it could not be studied further.
- Four publications incorrectly claimed to have conducted case study research, but an analysis of the papers determined no empirical work was presented.
- One publication was found to be erroneously included within the scope of the study. This publication addresses quality trade-offs for physical products using approaches that have been applied in the software development industry.

The results for *Phase 1* were updated for the publication that was erroneously included in the study. There is no information to suggest that the paper for which the full-text was not available was incorrectly classified during *Phase 1*. To ensure a consistent approach was taken for all papers in *Phase 1*, the papers found to have erroneously reported empirical studies were not reclassified.

### 3.6. Validity threats

The most significant threats to validity with a systematic map are bias in the selection of publications and the data extraction process [20]. To address these risks, a research protocol was defined prior to conducting the research. The research protocol defined the research questions to be answered, the databases to be used, the search strings to be used, the inclusion and exclusion criteria to be used, the resolution process where reviewers disagreed and the data extraction process.

One of the risks faced by researchers conducting this type of research in the software engineering domain is the lack of standard language and terminology [20]. To reduce this risk keywords were identified in an iterative process, using thesauri and publications to identify synonyms that should be included in the list of keywords. This activity was undertaken with the support of librarians specialising in software engineering.

Each paper was reviewed by at least two authors against the inclusion and exclusion criteria to reduce the risk of a publication being incorrectly included or excluded from the systematic map. Where the authors had conflicting reviews, a third author was required to review the publication and make a final decision.

It is possible for authors to introduce bias during the data extraction process. To reduce this risk, the authors of this study based the data extraction on the words used in each publication wherever possible.

This approach generated another threat to validity, with some authors incorrectly using terms leading to the incorrect classification of publications [47]. Four such situations were identified in *Phase 2* of this research. The results of *Phase 1* were not updated based on this result. This study does not aim to criticise the peer review publication process, and updating records based on an analysis only applied to a subset of the publications from *Phase 1* risked introducing systematic biases. Further, a number of publications did not contain sufficient data to independently verify claims made by the authors, making it impossible to guarantee correct classification in these cases. Thus the authors referred to the information as presented from the peer reviewed publication process.

## 4. Phase 1: Results and discussion of all publications

In total 168 publications identified and analysed as part of the systematic map.

Splitting the publications by the year in which they were published show a general increasing trend in the number of publications, peaking in 2009 with 42 publications. A breakdown of these results is shown in Fig. 2. Comparing the 26 publications for 2010 against the 41 publications for 2009 provides insufficient data to know if this is the start of a downward trend or is an anomaly in the results.

### 4.1. Key publication venues

The 169 publications identified as part of the systematic map were published at 121 different venues. Conferences were by far the most common venue type with 106 publications, followed by journals with 38 publications and then workshops with 24 publications.

The publication venues with more than two identified publications from the systematic map are listed in Table 4. This list of venues covers 24% of the identified publications. The venues include six conferences, four journal and one workshop.

All of the venues with more than two identified publications are focused on various aspects of software engineering. Some venues, like the *Software Quality Journal* and the *International Working Conference on Requirements Engineering*, focus on software quality. However, none of the venues are focused specifically on trade-offs within software engineering.

The list of venues presented in Table 4 are all highly ranked in different fora. With the exception of the *International Working Conference on Requirements Engineering*, all these venues are or have been listed in the *ISI Web of Knowledge*, although not all years of all conferences and workshops are listed. Further, all venues are listed in the Australian Core Ranking for Information System, except for *ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*.

### 4.2. Key researchers

The 168 publications included in the systematic map list 485 authors, which corresponds to 399 people. Most people were only listed on one publication, but 64 people were listed on two or more publications. The list of authors listed on three or more publications is presented in Table 5. with the number of publications with which they are associated from the systematic map.

The results show a number research collaborations. Berntsson Svensson and Regnell coauthored all six of the publications listed under their name; Reussner and Martins coauthored four publications; Huang and Mei coauthored three publications; Barney and Wohlin coauthored two publications; Davidsson and Svahnberg coauthored two publications; and Svahnberg and Wohlin coauthored one publication.

### 4.3. Research approach and focal development artefacts

This section breaks the 168 identified publications by the research approach taken and development artefact that is the focus of the publication. The classification of research approach is done using the classification system of Wieringa et al. [61]. Publications were also linked to the development artefact to it most strongly related—process, requirements, architecture, code, testing or runtime. The results of this analysis are presented visually in Fig. 3.

Within recent software quality trade-off research, *solution proposals* stand out as the dominant research approach. This approach is used in 61% of the identified publications. By contrast, the second

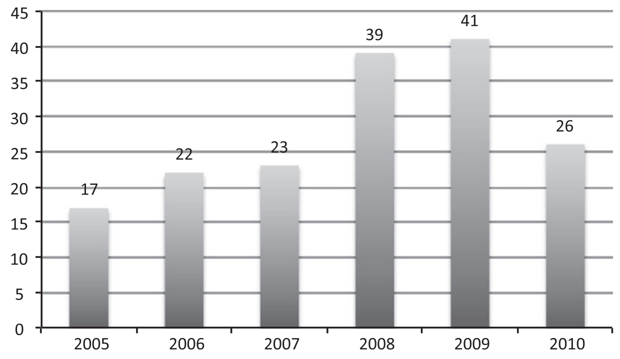

**Fig. 2.** Number of publications per year studied.

**Table 5**
Key researchers.

| Rank | Author | Publications |
|---|---|---|
| 1 | Berntsson Svensson, Richard | 6 |
| 1 | Regnell, Björn | 6 |
| 3 | Reussner, Ralf | 5 |
| 4 | Martens, Anne | 4 |
| 5 | Barney, Sebastian | 3 |
| 5 | Daneva, Maya | 3 |
| 5 | Davidsson, Paul | 3 |
| 5 | Huang, Gang | 3 |
| 5 | Lee, Dan Hyung | 3 |
| 5 | Mei, Hong | 3 |
| 5 | Ramdane-Cherif, Amar | 3 |
| 5 | Svahnberg, Mikael | 3 |
| 5 | Wohlin, Claes | 3 |

**Table 4**
Venues with more than two identified publications.

| Type | Venue | Publications |
|---|---|---|
| Conference | ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing | 5 |
| Journal | Information and Software Technology | 4 |
| Journal | Journal of Systems and Software | 4 |
| Journal | Software Quality Journal | 4 |
| Conference | Asia–Pacific Software Engineering Conference | 4 |
| Conference | International Conference on Software Engineering | 4 |
| Conference | International Working Conference on Requirements Engineering | 4 |
| Journal | IEEE Software | 3 |
| Conference | Empirical Software Engineering and Measurement | 3 |
| Conference | Euromicro—Software Engineering and Advanced Applications | 3 |
| Workshop | International Workshop on Software Product Management | 3 |

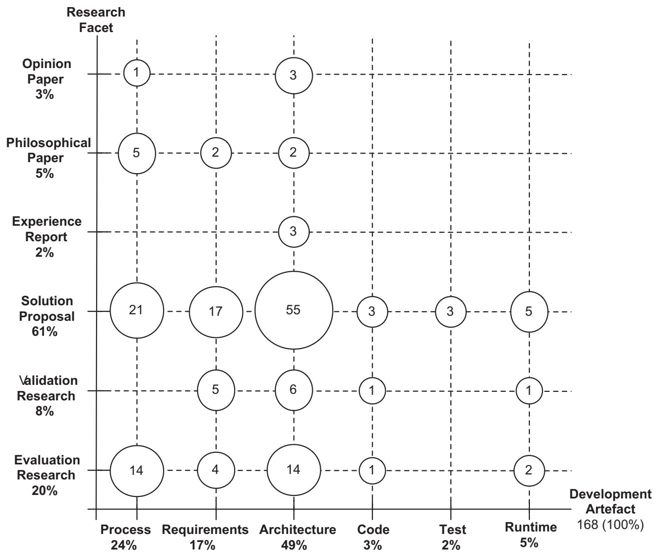

**Fig. 3.** Number of publications per research facet and development artefact.

most common research approach approach used is *evaluation research*, covering 20% of the publications. Of the remaining publications, 8% employ *validation research*, 5% present *philosophical papers*, 3% present *opinion papers* and 2% present *experience reports*.

When breaking down the papers by development artefact, another strong tendency is observed. At 49%, almost half of the identified publications look at software quality trade-offs within the architecture domain. The second most common development artefact address by recent software quality trade-off is the software development *process*, with 24% of publications, followed by *requirements* engineering with 17% of publications. Little attention has been given to software quality trade-offs at *runtime*, in *code* and during *testing*, with 5%, 3% and 2% of the identified publications respectively addressing these development artefacts.

Looking at the identified publications in terms of both the research approach and the development artefact provides some interesting insights. Given the clear focus on the *architecture* development artefact, one might expect a greater level of maturity with research, with a higher proportion of empirical research. This was not the case, however, with only 24% of publications addressing the *architecture* development aspects employing *validation* or *evaluation* research. The comparison values for the *process* and *requirements* development artefacts are 36% and 32% respectively.

## 4.4. Quality aspects studied

Publications were classified according the aspects of software quality covered in the abstract. A break-down of these results is provided in Table 6.

From the abstracts, most papers address software quality aspects generally—this includes aspects of quality (e.g. ISO 9126) and quality requirements (e.g. boot time must be less than three seconds). For most publications it was not possible to ascertain which specific qualities were studied from the abstract. Quality aspects and quality requirements were combined as it was not possible to differentiate this level of detail from the abstracts given the range and manner in which terms used to describe software quality. Papers specifically mentioning ISO 9126 in the abstract were separated from this group, with 13 publication identified (8%) using this quality model specifically.

Requirements more generally, and specifically including quality requirements, are address in 23 publications (14%).

In total 9 publications (5%) addressed specific pairs of quality. Only the *performance/reliability* pair was studied more than once, with two publications. However, *performance* was the most commonly studied quality aspects in this group, included in five publications.

Specific groups of quality were studied in four publications (2%). Of these two addressed groups of quality aspects relating to *security*, with the others addressing different sets of aspects.

**Table 6**
Quality aspects studied.

| Quality aspects | Publications |
|---|---|
| Software quality | 98 |
| Requirements | 23 |
| Quality requirements | 21 |
| ISO 9126 | 13 |
| Specific pair | 9 |
| Specific group | 4 |

**Table 7**
Approaches studied.

| Approach | Publications |
|---|---|
| Analytical Hierarchy Process (AHP) based | 22 |
| Model-based | 22 |
| Multiple | 11 |
| Algorithmic-based | 9 |
| Architecture Tradeoff Analysis Method (ATAM) based | 8 |
| Metrics-based | 7 |
| Expert opinion | 5 |
| QFD-based | 5 |
| Prototyping | 4 |

## 4.5. Trade-off approaches

A very wide array of trade-off approaches have been employed in the identified publications. The most commonly employed approaches are listed in Table 7, with the corresponding number of publications. This list covers 55% of the identified publications.

However, this list presents a simplified view of the approaches taken. For example, within the model-based approaches different models are used (e.g. UML, other architectural models) and the models are used in different ways to make the trade-off. Similarly the AHP-based trade-off approaches varied in their implementation. Expanding out the list of approaches to identify each unique approach would result in almost as many approaches as identified publications.

A selection of some of the approaches used in the other papers include an agent-based approach, artificial intelligence (AI) based approaches, game theory, a goal-question-metric (GQM) based approach, optimisation theory, Pareto curves, a perspective-based approach, a risk-driven approach, search-based techniques, utility theory and various statistical methods.

## 4.6. Systematic mapping discussion

The greatest surprise for the authors in conducting this research was the range of trade-off approaches used in the publications identified. While the authors had hoped to develop a classification of different approaches for software quality trade-offs, this proved neither feasible nor useful give the diverse range. This result suggests that there is an immaturity in the field in that no trade-off approach or set of approaches have emerged as candidates to dominate the research space.

While it is positive to see such a wide range of trade-off approaches are being considered, it creates concern that a clear majority of the research (61%) is classified as *solution proposals*. By contrast only 28% of the publications identified present empirically validated results, either *validation research* or *evaluation research*. While it is important in immature fields to consider a wide range of options, the lack of empirical research means only limited comparisons can be made between the strengths and weaknesses of the various techniques. More information on how each method performs in practice would allow for greater focus on the methods displaying greater strength. Ultimately, the aim of any engineering should be to develop and refine something of use.

The next question is to ask why so many of the *solution proposals* are not being submitted to empirical validation. There are a number of possible reasons for this result. Firstly, it is easier to propose a solution, than to propose and validate/evaluation a solution. However, methods that could be of value should be used, confirming if indeed this is the case. Secondly, it is generally easier to publish positive results than negative ones. This situation is unfortunate, as negative results would allow researchers and practitioners alike to better navigate the set of *solution proposals*.

Another interesting result is the very dominant focus of the quality trade-off approaches on the earlier phases of software development—process, requirements and architecture. This result is logical, as the earlier trade-offs are identified and made, the greater the range of options for dealing with the trade-offs in question. Further, problems are very costly to fix later. For example, if the architecture is wrong, then trying to fix the problem at runtime may not be possible.

## 5. Phase 2: Results and discussion for empirical publications

This section presents an overview of the publications that present empirical results towards addressing software quality trade-offs for the process and requirements artefacts—addressing RQ2. While *Phase 1* identified 48 empirical publications, only 43 are the subject of study in this section. Four publications were incorrectly classified by their authors as containing empirical work, and the full-text of one publication could not be sourced for further review.

The remainder of this section presents the results and discussion in terms of the trade-off approaches studied, the aims of the research, and the academic rigour and industrial relevance of the research.

### *5.1. Trade-off approaches*

The first aim of *Phase 2* was to assess the software quality trade-off approaches that have empirically studied. A summary of these results is provided in Table 8, broken down by development artefact.

The results show a diverse range of approaches to address software quality trade-offs with respect to the software development process and requirements. AHP is the most widely applied approach, being used by three different groups of researchers in three different contexts. The QUPER model is empirically assessed in an equal number of publications, but all this work has been done by the same group of researchers and two of the papers study a similar context.

Similarly a diverse range of approaches has been employed to address the architecture artefact. AHP is the only method found to be reused between multiple artefacts—process, requirements and architecture. However, there are similarities between a number of methods. The most used empirically research approaches focusing on architecture are automated generation of alternative architectures, SOA service pool management, ATAM and AHP.

AHP is further proposed for use in selecting algorithms by one of the paper classified as applying to the code and runtime artefacts.

None of the approaches stands out as dominating the research landscape, although AHP appears to be the most robust method, used against a number of development artefacts. In the absence of a clear direction it appears that researchers are looking for new and improved solution to empirically validate. However, the two publications that assess software quality trade-offs in different software packages for the procurement process use the AHP method.

There appear to be a number of common elements in the approaches used for software quality trade-offs. For the process and requirements artefacts a number of papers cite the need to create or tailor a quality model to a specific context before it can be used, and many papers also highly recommend the involvement of a diverse range of stakeholders when developing or tailoring a quality model and making trade-off decisions. Many groups are affected, and each in different ways—so by bringing everyone together it helps ensure a more mutually acceptable goal.

For the architecture artefacts, the general rules are to define the goals that need to be achieved, and then assess the ability for various architectures to meet these goals. This work should be used to

**Table 8**
Publications by development artefact and trade-off approach.

| Artefact | Approach | Reference |
|---|---|---|
| Process | AHP-based | [14] |
| | | [59] |
| | Automated negotiaton | [45] |
| | SAAM-SQ | [2] |
| | | [1] |
| | Expert opinion | [60] |
| | Metrics | [34] |
| | Model-based | [62] |
| | Ranking | [54] |
| Requirements | AHP-based | [37] |
| | | [44] |
| | Approaches used in practice | [5] |
| | QUPER Model | [51] |
| | | [4] |
| | | [7] |
| | Negotiation/collaboration | [36] |
| | Networked systems survivability | [41] |
| | Prospect theory | [21] |
| | Prototyping | [43] |
| | QFD-based | [48] |
| Architecture | AHP-based | [17] |
| | Algorithmic | [27] |
| | | [38] |
| | | [40] |
| | | [55] |
| | ATAM-based | [10] |
| | | [39] |
| | Expert opinion | [56] |
| | Goal structuring notation | [3] |
| | Metrics | [11] |
| | Model-based | [16] |
| | | [64] |
| | Multiple | [23] |
| | | [28] |
| | Prototyping | [58] |
| | Replication | [63] |
| | Search-based | [19] |
| | SPE | [18] |
| Code | Algorithmic optimisation | [42] |
| | Metrics | [35] |
| Runtime | Algorithmic | [25] |
| | | [49] |
| | | [50] |

**Table 9**
Approaches applied in the research from each phase.

| Approach | Phase 1 (%) | Phase 2 (%) |
|---|---|---|
| AHP | 13 | 12 |
| Model-based | 13 | 14 |
| Multiple | 7 | 5 |
| Algorithmic-based | 5 | 14 |
| ATAM-based | 5 | 5 |
| Metrics-based | 4 | 7 |
| Expert opinion | 3 | 5 |
| QFD-based | 3 | 2 |
| Prototyping | 2 | 2 |

**Table 10**
Rigour and relevance classification system.

| Area | Aspect | Acronym |
|---|---|---|
| Rigour | Context described | Ri1 |
| | Study design described | Ri2 |
| | Validity discussed | Ri3 |
| Relevance | Subjects | Re1 |
| | Context | Re2 |
| | Scale | Re3 |
| | Research method | Re4 |

think creatively about alternative solution that may better meet the needs of the system under development.

The research also highlights that any trade-off decision is very particular to its context. Embedded systems face different challenges to web-based systems and combat management systems. This emphasis the need for greater empirical work in this area, as methods will need to be robust in dealing with such a diverse range of contexts.

It is also possible to compare the software quality trade-off approaches used in publications from *Phase 1* with those empirically assessed in *Phase 2*. The results, presented in Table 9, show a very similar distribution for the most studied approaches. The only notable exception is that only 5% of publications from *Phase 1* use an algorithmic-based approach, while 14% of the publication from *Phase 2* use such an approach. It is possible that algorithmic-based approaches are relatively easier to assess empirically, and thus are relatively overrepresented in *Phase 2*.

## *5.2. Aim of publications*

The second aim was to determine the aim of researchers conducting empirical software quality trade-off research. The results show a diverse range of research objectives within the research space surveyed. The most commonly cited objectives for the process and requirements artefacts were to determine the priorities on the qualities to better understand the quality requirements. Some papers sought to take this further, by using this information to provide improvement suggestions, or put quality requirements on the development roadmap.

Most publications focusing on the architecture artefacts aimed to provide general approaches to assist in the assessment of various architectures in their ability to meet different quality goals. However, a small number of papers presented finding from specific cases researchers had sought to resolve.

The papers addressing the code and runtime artefacts were much more technical in nature—mainly providing or testing algorithms to support software quality trade-offs.

## *5.3. Academic rigour and industrial relevance*

Each paper included in *Phase 2* was given a rating according to its rigour and relevance, as defined by Ivarsson and Gorschek [30].

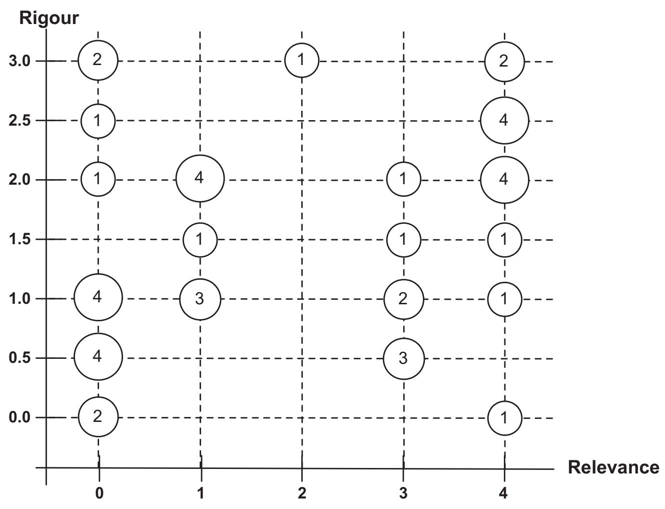

**Fig. 4.** Rigour and relevance evaluation of empirical publications.

**Table 11**
Rigour and relevance classification of empirical publications.

| Paper | Rigour | | | Relevance | | | | Totals | |
|---|---|---|---|---|---|---|---|---|---|
| | Ri1 | Ri2 | Ri3 | Re1 | Re2 | Re3 | Re4 | Ri | Re |
| [2] | 0.5 | 1.0 | 1.0 | 1 | 1 | 1 | 1 | 2.5 | 4 |
| [1] | 0.5 | 1.0 | 0.0 | 1 | 1 | 1 | 1 | 1.5 | 4 |
| [3] | 0.0 | 1.0 | 0.0 | 0 | 0 | 0 | 0 | 1.0 | 0 |
| [4] | 1.0 | 1.0 | 1.0 | 1 | 1 | 1 | 1 | 3.0 | 4 |
| [5] | 0.5 | 1.0 | 1.0 | 1 | 1 | 0 | 1 | 2.5 | 3 |
| [7] | 1.0 | 1.0 | 1.0 | 1 | 1 | 1 | 1 | 3.0 | 4 |
| [10] | 1.0 | 1.0 | 0.5 | 1 | 1 | 1 | 1 | 2.5 | 4 |
| [11] | 0.5 | 0.5 | 0.5 | 0 | 0 | 1 | 0 | 1.5 | 1 |
| [16] | 0.5 | 0.5 | 0.0 | 0 | 0 | 1 | 0 | 1.0 | 1 |
| [14] | 1.0 | 0.5 | 0.0 | 0 | 1 | 1 | 1 | 1.5 | 3 |
| [17] | 1.0 | 1.0 | 0.5 | 1 | 1 | 1 | 1 | 2.5 | 4 |
| [18] | 1.0 | 1.0 | 0.5 | 1 | 1 | 1 | 1 | 2.5 | 4 |
| [19] | 0.5 | 0.5 | 0.0 | 1 | 1 | 1 | 0 | 1.0 | 3 |
| [21] | 1.0 | 1.0 | 1.0 | 0 | 0 | 0 | 0 | 3.0 | 0 |
| [23] | 1.0 | 1.0 | 0.5 | 0 | 0 | 0 | 0 | 2.5 | 0 |
| [25] | 0.5 | 0.5 | 0.0 | 0 | 0 | 0 | 0 | 1.0 | 0 |
| [27] | 0.5 | 0.5 | 0.0 | 0 | 0 | 0 | 0 | 1.0 | 0 |
| [28] | 0.0 | 0.5 | 0.0 | 0 | 0 | 0 | 0 | 0.5 | 0 |
| [34] | 0.5 | 0.0 | 0.5 | 1 | 1 | 1 | 1 | 1.0 | 4 |
| [35] | 0.5 | 1.0 | 0.5 | 0 | 0 | 0 | 0 | 2.0 | 0 |
| [36] | 0.5 | 1.0 | 0.0 | 1 | 1 | 0 | 1 | 1.5 | 3 |
| [37] | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0.0 | 0 |
| [38] | 0.0 | 0.5 | 0.0 | 0 | 0 | 0 | 0 | 0.5 | 0 |
| [39] | 0.5 | 0.5 | 0.0 | 0 | 0 | 1 | 0 | 2.0 | 1 |
| [40] | 0.5 | 0.5 | 1.0 | 0 | 0 | 1 | 0 | 2.0 | 1 |
| [41] | 0.5 | 0.5 | 0.0 | 0 | 1 | 1 | 1 | 1.0 | 3 |
| [42] | 0.5 | 0.5 | 0.0 | 0 | 0 | 1 | 0 | 2.0 | 1 |
| [43] | 1.0 | 1.0 | 0.0 | 0 | 0 | 1 | 0 | 2.0 | 1 |
| [44] | 1.0 | 1.0 | 0.0 | 1 | 1 | 1 | 1 | 2.0 | 4 |
| [45] | 0.5 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0.5 | 0 |
| [48] | 0.5 | 1.0 | 0.0 | 1 | 0 | 1 | 1 | 1.5 | 3 |
| [49] | 0.5 | 0.5 | 0.0 | 0 | 0 | 1 | 0 | 1.0 | 1 |
| [50] | 0.5 | 0.5 | 0.0 | 0 | 0 | 1 | 0 | 1.0 | 1 |
| [51] | 0.5 | 0.5 | 0.0 | 1 | 1 | 0 | 1 | 1.0 | 3 |
| [54] | 0.5 | 1.0 | 0.5 | 1 | 1 | 1 | 1 | 2.0 | 4 |
| [55] | 0.5 | 0.5 | 0.0 | 0 | 0 | 0 | 0 | 1.0 | 0 |
| [56] | 1.0 | 1.0 | 1.0 | 0 | 0 | 0 | 0 | 3.0 | 0 |
| [58] | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0.0 | 0 |
| [59] | 1.0 | 1.0 | 0.0 | 1 | 1 | 1 | 1 | 2.0 | 4 |
| [60] | 0.5 | 0.5 | 1.0 | 1 | 1 | 1 | 1 | 2.0 | 4 |
| [62] | 0.0 | 0.0 | 0.0 | 1 | 1 | 1 | 1 | 0.0 | 4 |
| [63] | 0.5 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0.5 | 0 |
| [64] | 0.5 | 0.5 | 0.0 | 0 | 0 | 1 | 1 | 3.0 | 2 |
| Totals | 24.5 | 28.0 | 12.0 | 18 | 19 | 26 | 20 | 68.5 | 83 |

Rigour refers to the the extent and detail that context is explained in the publication. It is possible for a publication to score poorly if the research was rigorous, but insufficient information was detailed in the publication. Relevance evaluates the potential for the results to impact industry by considering the realism of the scenario used for evaluation. The approach proposed in Ivarsson and Gorschek [30] awards points for achieving certain criteria that define rigour and relevance, with a maximum of three points for rigour and four points for relevance. A summary of the model is provided in Table 10.

The classification of papers from *Phase 2* are summarised in Fig. 4, and detailed in Table 11 using the acronyms in Table 10.

These results are positive, showing significantly higher levels of rigour and relevance than are seen in Ivarsson and Gorschek [30]. However, there is still a large range in these results.

Of the areas assessed using this method, one stood out as falling short in comparison to the other aspects. *Validity* was usually not discussed in the publications.

While it was positive to find higher levels of research rigour and relevance than was expected by the authors, the publications are by no means perfect. An attempt was made to classify the research against the model of context in industrial software engineering research proposed by Petersen and Wohlin [46]. Such a classification would provide much detail into the contexts that have been studied and the areas where further work is required. Unfortunately so much information was not published in the papers that no meaningful result could be obtained on the contexts that have been studied and those that have not.

As previously mentioned, discussions of research validity were commonly missing or lacked sufficient detail.

## 6. Conclusion

This paper seeks to answer two research questions with respect to software quality trade-offs.

The first research question, RQ1, aims to identify the state of the art with respect to software quality trade-off literature. A systematic map of 168 publications has been conducted to answer this question. The map covered the key research venues, key researchers, research approaches applied, development artefacts on which trade-offs are applied and the trade-off approaches studied.

A clear majority of the research is focused in the early phases of the development cycle—with 90% of the publications address process, requirements and architecture. Architecture stands out as the artefact on which most software quality trade-off research is undertaken, with 49% of the identified publications focusing on this artefact.

The results show the software quality trade-off research area to still be maturing, with 61% of the research providing non-empirically assessed solution proposals. Only 28% of the publications provide empirical evidence. Further, a very diverse range of approaches for conducting software quality trade-offs are still being proposed and explored. AHP is the most research approach, employed in 13% of the publications. Model-based approaches are equally applied, but vary greatly in their implementation.

The quality trade-off approaches cover a wide solution space. Researchers are drawing from computer science, software engineering, economics, mathematics and statistics to find suitable approaches. However, most research on an approach seems to stop once the solution has been proposed. In order to be able to confirm the viability and compare these approaches, empirical research must be conducted. Without empirical research, practitioners and researchers alike are unable to determine which approach is the most suitable for a given context.

The second research question, RQ2, seeks to provide a more detailed overview with respect to empirical research addressing software quality trade-offs. *Phase 1* of the research identified 48 publications within the scope of this question. After reading and classifying the publications only 43 were found to be within scope of *Phase 2*.

The aims for conducting software quality trade-offs are varied. When looking at papers addressing the process and requirements artefacts, most aim to help elicit priorities, which can then be explicitly or implicitly turned into requirements. Some publications emphasised the need for a common understanding to achieve software quality success, with one claiming that knowledge of the priorities on quality can provide a sufficient understanding of the quality requirements.

The results for the architecture artefact also showed a diverse range of approaches are being used to assess the ability for a software architecture to meet quality goals, with a number trying to improve the architecture before the software is built.

As with *Phase 1*, the results of *Phase 2* highlight that there is no clear approach used for software quality trade-offs. *Phase 2* emphasises this point with respect to the process and requirements artefacts.

However, there are some commonalities between the various approaches to software quality trade-offs. Many publications emphasise the need to create a model of software quality tailored to the context in which it is being applied, and to involve a diverse range of perspectives in any trade-off process to help ensure all aspects of the product development are properly considered. Further, AHP is the trade-off approach of choice when choosing between software packages as part of the procurement process.

AHP is also found to be the most robust approach, being applied or proposed for use with regard to all development artefacts.

This research found a very wide vocabulary used to describe software quality. In using such a wide research vocabulary, researchers make it difficult to people find relevant publications. Going forward it would be advantageous to develop a standard vocabulary to describe software quality to help overcome these issues.

Further, the authors would like to recommend authors who develop solution proposals to name these proposals. This allows different approaches to be quickly and easily identified. Without a name for solution proposals, it can become very difficult to search for reviews, validation and evaluation of this work.

Finally, the results highlight the need for greater empirical research. While a wide range of solutions are being proposed, the lack of empirical evidence means only limited comparisons and evaluations can be made between the methods. Without this understanding it not possible to determine and transfer best-practice to industrial practitioners.

## Acknowledgements

This work was partly funded by *Vinnova*, *Sony Ericsson Mobile Communications*, *Ericsson*, *ST-Ericsson*, *Axis Communications* and *Softhouse Consulting Baltic* as part of the Industrial Excellence Center for Embedded Applications Software Engineering (EASE).

## References

[1] Sebastian Barney, Claes Wohlin, Software product quality: Ensuring a common goal, in: Qing Wang, Ray Madachy, Dietmar Pfahl (Eds.), Proceedings of the International Conference on Software Process (ICSP), 2009, pp. 256–267.

[2] Sebastian Barney, Claes Wohlin, Aybüke Aurum, Balancing software product investments, in: Empirical Software Engineering and Management (ESEM), 2009, pp. 257–268.

[3] I. Bate, Systematic approaches to understanding and evaluating design trade-offs, Journal of Systems and Software 81 (8) (2008) 1253–1271.

[4] Richard Berntsson Svensson, Thomas Olsson, Björn Regnell, Introducing support for release planning of quality requirements: an industrial evaluation of the QUPER model, in: Second International Workshop on Software Product Management (IWSPM), 2008, pp. 18–26.

[5] Richard Berntsson Svensson, Tony Gorschek, Björn Regnell, Quality requirements in practice: An interview study in requirements engineering for embedded systems, in: Proceedings of 15th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), LNCS, vol. 5512, Amsterdam, Netherlands, June 2009, pp. 218–232.

[6] Richard Berntsson Svensson, Martin Höst, Björn Regnell, Managing quality requirements: A systematic review, in: 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), 2010a, pp. 261–268.

[7] Richard Berntsson Svensson, Yuri Sprockel, Björn Regnell, Sjaak Brinkkemper, Cost and benefit analysis of quality requirements in competitive software product management: A case study on the quper model, in: Fourth International Workshop on Software Product Management (IWSPM), 2010b, pp. 40–48.

[8] Stefan Biffl, Aybüke Aurum, Barry Boehm, Hakan Erdogmus, Paul Grünbacher (Eds.), Value-Based Software Engineering, Springer, Berlin Heidelberg, 2006.

[9] Barry Boehm, Apurva Jain, An initial theory of value-based software engineering, in: Stefan Biffl, Aybüke Aurum, Barry Boehm, Hakan Erdogmus, Paul Grünbacher (Eds.), Value-Based Software Engineering, Springer, Berlin Heidelberg, 2006, pp. 15–37.

[10] Nelis Boucké, Danny Weyns, Kurt Schelfthout, Tom Holvoet, Applying the ATAM to an architecture for decentralized control of a transportation system, in: Christine Hofmeister, Ivica Crnkovic, Ralf Reussner (Eds.), Quality of Software Architectures, Lecture Notes in Computer Science, vol. 4214, Springer, Berlin/Heidelberg, 2006, pp. 180–198.

[11] K. Buyens, R. Scandariato, W. Joosen, Measuring the interplay of security principles in software architectures, in: 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM), 2009, pp. 554–563.

[12] Yolande E. Chan, Why haven't we mastered alignment? The importance of the informal organization structure, MIS Quarterly Executive 1 (2) (2002) 97–112.

[13] Yolande E. Chan, Blaize Horner Reich, IT alignment: an annotated bibliography, Journal of Information Technology 22 (4) (2007) 316–396.

[14] Che-Wei Chang, Cheng-Ru. Wu, Hung Lin Lin, Integrating fuzzy theory and hierarchy concepts to evaluate software quality, Software Quality Journal 16 (2) (2008) 263–276.

[15] Lawrence Chung, Brian A. Nixon, Eric Yu, John Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer Academic, 2000.

[16] Vittorio Cortellessa, Catia Trubiani, Leonardo Mostarda, Naranker Dulay, An architectural framework for analyzing tradeoffs between software security and performance, in: H Giese (Ed.), Proceedings Architecting Critical Systems, Lecture Notes in Computer Science, vol. 6150, 2010, pp. 1–18. 1st International Symposium on Architecting Critical Systems, Prague, Czech Republic, June 23–25, 2010.

[17] Mahsa Razavi Davoudi, Fereidoon Shams Aliee, A new AHP-based approach towards enterprise architecture quality attribute analysis, in: Third International Conference on Research Challenges in Information Science (RCIS), Fez, Morocco, 2009, pp. 333–342.

[18] C. Del Rosso, Software performance tuning of software product family architectures: Two case studies in the real-time embedded systems domain, Journal of Systems and Software 81 (1) (2008) 1–19.

[19] J. D́iaz-Pace, Marcelo Campo, Using planning techniques to assist quality-driven architectural design exploration, in: Sven Overhage, Clemens Szyperski, Ralf Reussner, Judith Stafford (Eds.), Software Architectures, Components, and Applications, Lecture Notes in Computer Science, vol. 4880, Springer, Berlin/Heidelberg, 2007, pp. 33–52.

[20] Tore Dybå, Torgeir Dingsøyr, Empirical studies of agile software development: a systematic review, Information and Software Technology 50 (9–10) (2008) 833–859.

[21] Nina Fogelström, Sebastian Barney, Aybüke Aurum, Anders Hederstierna, When product managers gamble with requirements: attitudes to value and risk, Requirements Engineering: Foundation for Software Quality (2009) 1–15.

[22] David A. Garvin, What does "product quality" really mean?, Sloan Management Review 26 (1) (1984) 25–43

[23] Anna Grimán, Maria Pérez, L. Mendoza, F. Losavio, Feature analysis for architectural evaluation methods, Journal of Systems and Software 79 (6) (2006) 871–888.

[24] Georg Herzwurm, Sixten Schockert, Wolfram Pietsch, QFD for customer-focused requirements engineering, in: Proceedings of the 11th IEEE International Requirements Engineering Conference, 2003, pp. 330–338.

[25] Ma Hongwei, Zhao Xiumei, An efficiency and fairness based packet marking algorithm, in: 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing (SNPD), 2009, pp. 407–410.

[26] Robert W. Hoyer, Brooke B.Y. Hoyer, What is quality?, Quality Progress 34 (7) (2001) 53–62

[27] Gang Huang, Li Zhou, Xuan-Zhe Liu, Hong Mei, Shing-Chi Cheung, Performance aware service pool in dependable service oriented architecture, Journal Computer Science Technology 21 (4) (2006) 565–573.

[28] Gang Huang, Xuanzhe Liu, Hong Mei, SOAR: Towards dependable service-oriented architecture via reflective middleware, International Journal of Simulation and Process Modelling 3 (1–2) (2007) 55–65.

[29] ISO9126, Software engineering – product quality – part 1: Quality model, International Standards Organization, 2001.

[30] Martin Ivarsson, Tony Gorschek, A method for evaluating rigor and industrial relevance of technology evaluations, Empirical Software Engineering (2010) 1–31.

[31] Enrico Johansson, Anders Wesslén, Lars Bratthall, Martin Höst, The importance of quality requirements in software platform development-a survey, in: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS), 2001.

[32] Barbara Kitchenham, Guidelines for performing systematic literature reviews in software engineering, Technical Report EBSE200701, Keele University and Durham University, 2007.

[33] Barbara Kitchenham, Shari Lawrence Pfleeger, Software quality: The elusive target, IEEE Software 13 (1) (1996) 12–21.

[34] Rogerio T.O. Lacerda, Leonardo Ensslin, Sandra R. Ensslin, A study case about a software project management success metrics, in: 33rd Annual IEEE Software Engineering Workshop (SEW), 2009, pp. 45–54.

[35] Niklas Lavesson, Paul Davidsson, Quantifying the impact of learning algorithm parameter tuning, in: Proceedings of the 21st National Conference on Artificial Intelligence, vol. 1, 2006, pp. 395–400.

[36] Giovana Linhares, Marcos Borges, Pedro Antunes, Negotiation-collaboration in formal technical reviews, in: Luís Carriço, Nelson Baloian, Benjamim Fonseca (Eds.), Groupware: Design, Implementation, and Use, Lecture Notes in Computer Science, vol. 5784, Springer, Berlin/Heidelberg, 2009, pp. 344–356.

[37] Xiaojing Liu, Jihong Pang, A fuzzy synthetic evaluation method for software quality, in: 2nd International Conference on e-Business and Information System Security (EBISS), May 2010, pp. 1–4.

[38] Yuan-sheng Luo, Yong Qi, Lin-feng Shen, Di Hou, C. Sapa, Ying Chen, An improved heuristic for QoS-aware service composition framework, in: 10th IEEE International Conference on High Performance Computing and Communications (HPCC), 2008, pp. 360–367.

[39] F. Mahananto, R.P. Wibowo, Evaluation on organic web based software architecture of healthcare information system, in: 40th International Conference on Computers and Industrial Engineering (CIE), 2010, pp. 1–6.

[40] Anne Martens, Heiko Koziolek, Steffen Becker, Ralf Reussner, Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms, in: Proceedings of the First Joint WOSP/SIPEW International Conference on Performance Engineering, WOSP/SIPEW '10, 2010, p. 105.

[41] Nancy R. Mead, Ted Stehney, Security quality requirements engineering (SQUARE) methodology, ACM SIGSOFT Software Engineering Notes 30 (4) (2005) 1.

[42] S. Misailovic, S. Sidiroglou, H. Hoffmann, M. Rinard, Quality of service profiling, in: Proceedings – International Conference on Software Engineering, vol. 1, 2010.

[43] Marcio F.S. Oliveira, Ricardo Miotto Redin, Luigi Carro, Luís da Cunha Lamb, Flávio Rech Wagner, Software quality metrics and their impact on embedded software, in: 5th International Workshop on Model-based Methodologies for Pervasive and Embedded Software (MOMPES), 2008, pp. 68–77.

[44] Semih Onut, Tugba Efendigil, A theorical model design for erp software selection process under the constraints of cost and quality: a fuzzy approach, Journal of Intelligent and Fuzzy Systems 21 (6) (2010).

[45] Vikram Patankar, Rattikorn Hewett, Automated negotiations in web service procurement, in: Third International Conference on Internet and Web Applications and Services (ICIW), 2008, pp. 620–625.

[46] Kai Petersen, Claes Wohlin, Context in industrial software engineering research, in: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM '09, 2009, pp. 401–404.

[47] Kai Petersen, Robert Feldt, Shahid Mujtaba, Michael Mattsson, Systematic mapping studies in software engineering, in: 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2008, pp. 71–80.

[48] João Ramires, Pedro Antunes, Ana Resṕicio, Software requirements negotiation using the software quality function deployment, in: Hugo Fuks, Stephan Lukosch, Ana Carolina Salgado (Eds.), Groupware: Design, Implementation, and Use, Lecture Notes in Computer Science, vol. 3706, Springer, Berlin/Heidelberg, 2005, pp. 308–324.

[49] A.J. Ramirez, D.B. Knoester, B.H.C. Cheng, P.K. McKinley, Plato: a genetic algorithm approach to run-time reconfiguration in autonomic computing systems, Cluster Computing (2010).

[50] Andres J. Ramirez, Betty H.C. Cheng, Philip K. McKinley, Benjamin E. Beckmann, Automatically generating adaptive logic to balance non-functional tradeoffs during reconfiguration, in: ICAC, 2010b.

[51] Björn Regnell, Martin Höst, Richard Berntsson Svensson, A quality performance model for cost-benefit analysis of non-functional requirements applied to the mobile handset domain, in: Pete Sawyer, Barbara Paech, Patrick Heymans (Eds.), Requirements Engineering: Foundation for Software Quality, Lecture Notes in Computer Science, vol. 4542, Springer, Berlin/Heidelberg, 2007, pp. 277–291.

[52] Björn Regnell, Richard Berntsson Svensson, Thomas Olsson, Supporting roadmapping of quality requirements, IEEE Software 25 (2) (2008) 42–47.

[53] Mehwish Riaz, Muhammad Sulayman, Norsaremah Salleh, Emilia Mendes, Experiences conducting systematic reviews from novices' perspective, in: 14th International Conference on Evaluation and Assessment in Software Engineering (EASE), April 2010.

[54] Mbusi Sibisi, Cornelis Cristo Van Waveren, A process framework for customising software quality models, in: IEEE AFRICON Conference, 2007.

[55] A. Sil, O. Bandyopadhyay, N. Chaki, Data diverse fault tolerant architecture for component based systems, in: World Congress on Nature Biologically Inspired Computing (NaBIC), 2009, pp. 942–946.

[56] M. Svahnberg, C. Wohlin, An investigation of a method for identifying a software architecture candidate with respect to quality attributes, Empirical Software Engineering 10 (2) (2005) 149–181.
[57] Mikael Svahnberg, Claes Wohlin, Lars Lundberg, Michael Mattsson, A quality-driven decision-support method for identifying software architecture candidates, International Journal of Software Engineering and Knowledge Engineering 13 (5) (2003) 547–573.
[58] S. Tosun, N. Mansouri, E. Arvas, M. Kandemir, Y. Xie, W.-L. Hung, Reliability-centric hardware/software co-design, in: Sixth International Symposium on Quality of Electronic Design (ISQED), 2005, pp. 375–380.
[59] Jos Trienekens, Rob Kusters, Dennis Brussel, Quality specification and metrication, results from a case-study in a mission-critical software domain, Software Quality Journal 18 (2010) 469–490.
[60] Jari Vanhanen, Mika V. Mäntylä, Juha Itkonen, Lightweight elicitation and analysis of software product quality goals: A multiple industrial case study, in: Third International Workshop on Software Product Management (IWSPM), 2009, pp. 42–52.
[61] Roel Wieringa, Neil Maiden, Nancy Mead, Colette Rolland, Requirements engineering paper classification and evaluation criteria: A proposal and a discussion, Requirements Engineering 11 (2006) 102–107.
[62] Jamaiah Haji Yahaya, Aziz Deraman, Measuring unmeasurable attributes of software quality using pragmatic quality factor, in: 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 1, 2010, pp. 197–202.
[63] Gabriel L. Zenarosa, Soumya Simanta, Experiences in engineering active replication into a traditional three-tiered client-server system, in: Proceedings of the 2008 RISE/EFTS Joint International Workshop on Software Engineering for Resilient Systems, SERENE '08, 2008, pp. 55–60.
[64] Qian Zhang, Jian Wu, Hong Zhu, Tool support to model-based quality analysis of software architecture, in: 30th Annual International Computer Software and Applications Conference (COMPSAC), vol. 1, 2006, pp. 121–128.