A Framework to Support Software Quality Trade-offs from a Process-Based Perspective

Gabriel Alberto García-Mireles[1], M^a Ángeles Moraga[2], Félix García[2] and Mario Piattini[2]

[1]Departmento de Matemáticas, Universidad de Sonora
Blvd. Encinas y Rosales s/n col. Centro, 83000 Hermosillo, Sonora, México
mireles@gauss.mat.uson.mx

[2]Instituto de Tecnologías y Sistemas de Información, Universidad de Castilla-La Mancha, Paseo de la Universidad 4, 13071, Ciudad Real, España
{MariaAngeles.Moraga, Felix.Garcia, Mario.Piattini}@uclm.es

Abstract. Software organizations currently deploy various software quality frameworks simultaneously, which could have an impact on both product and process. From a product perspective, organizations are attempting to provide software that will meet stakeholders' quality requirements. However, experts in requirements have pointed out the complexity of dealing with quality requirements and their interactions, particularly those which are conflictive. Although various methods with which to carry out trade-off studies of software architecture models have been proposed, few have addressed the problem of software quality trade-offs from a process improvement perspective. Since a trade-off study is a kind of decision process, we have reviewed the decision processes in CMMI and ISO/IEC 12207 in order to identify the process requirements. As we wished to deal with only one set of requirements, we have applied a harmonization technique whose results show that it is possible to embed the ISO/IEC 12207 decision process into the CMMI decision process. With regard to the issues related to quality attribute interactions and the previously identified requirements, we have developed a proposal for a process framework to deal with these issues, which includes a trade-off quality process. We depict the elements taken into account to build the framework, and the trade-off process is presented at generic level.

Keywords: harmonization, mapping, trade-off study, CMMI-DEV, ISO/IEC 12207, quality requirements conflict, decision process.

1 Introduction

Software quality is a fundamental feature that must be addressed throughout the product development life cycle. Software quality is defined as "the extent in which software has a combination of desired attributes" [1]. These desired attributes can be found in software product quality models such as ISO/IEC 9126, FURPS, McCall, and others. Indeed, one important goal in software development is to achieve a balance among the desired quality attributes [2], but some of them may be very difficult, or even impossible, to implement in the software product when they contradict each other. This situation is termed as conflict [3]. Some of the factors that may cause conflict among quality attributes to arise are the individual's perception of quality [4], inconsistency among quality models [5], and the lack of appropriate methods and techniques [6]. Software engineers must consequently carry out a trade-off study in order to balance quality requirements and build a better system [1]. A trade-off study is described as being a systematic approach through which to analyze the advantages and disadvantages of each proposed requirement or design alternative [7].

Various studies have addressed the question of software quality trade-offs. For instance, the WinWin approach considers the conflicting requirements and uses a tool to inform relevant stakeholders about the possible strategies that can be deemed to resolve the situation [2]. The NFR framework provides a modeling approach in which quality requirements can be modeled as softgoals and the mechanisms proposed to achieve them can also be included [8]. In a recent mapping study, Barney et al. [9] found diverse solution proposals that can be used to tackle software quality trade-offs. The results showed that the majority of papers are focused on methods that support architecture trade-offs. The same authors also found evidence of trade-off methods applied to requirements and processes. At the process level, the principal topic is that of tailoring ISO9126 to a specific context and assigning weights to quality characteristics. As a conclusion, the authors pointed out that the research area is still maturing.

The literature reviewed depicts a number of methods and techniques for use in dealing with software quality tradeoffs, but their scope is limited to a particular research area. The aim of this paper is to propose a framework to deal with software quality tradeoffs at earlier software product development phases aligned with CMMI [10] and ISO12207 [11]. In previous work, we carried out a harmonization effort in order to identify practices that support product quality characteristics, and discovered that process improvement models address them in the analysis and design phases [12]. Since a trade-off study is a kind of decision process, we reviewed the requirements for the decision process from CMMI and ISO12207 by applying a harmonization approach [13]. The requirements identified have contributed to the process framework presented herein.
The process framework supports the tailoring of quality models in order to refine the understanding of quality terms with regard to the kind of software product that an organization develops. It also provides support to deal with interactions among quality characteristics, and when negative interactions are identified in the software project, the framework provides a software quality trade-off process.

The remainder of this paper is structured as follows: Section 2 describes the works existing in literature that concern the decision-making approach. In Section 3 the comparison between the CMMI and ISO12207 processes is presented. Section 4 depicts the framework proposed to deal with software quality trade-offs. Finally, Section 5 shows our conclusions and future work.

2 Related work

Many decision-making situations occur during software development. In practice, decision makers rely on their experience, attitude and intuition, and this depends on the context, such as the budget and the time available to make a decision [14]. In an empirical study concerning how software engineers make design decisions, Zannier et al. [15] reported that they can apply either a rational decision making approach when the problem under consideration is well-structured or a naturalistic decision making approach when the problem is perceived as ill-structured. Software project decisions can also be made at strategic, tactical and operational organizational levels [16].
Despite the importance of decision-making in software engineering, little empirical research has been reported [15]. Ruhe [17] summarizes the major concerns as regards decision-making and concludes that decisions are often poorly understood or described, made under time pressure, based on intuition, and consider only a few relevant stakeholders. Strategic and operational decisions concerning products, process, technologies or tools and other resources are far from being mature. Any stakeholder can perceive, interpret and evaluate the quality characteristics with regard to his/her own experience. This subjectivity could produce conflicting quality requirements [8]. Uncertainty and incompleteness are inherent characteristics of software quality requirements at the beginning of software development [18]. When conflicts emerge among quality requirements, software engineers should manage them. Indeed, as Robinson et al. [6] points out, requirements interaction management is a critical area.

Some proposals with which to manage conflicting quality requirements have also appeared, such as the WinWin approach [2], the NFR framework [8] and KAOS [19]. Various researchers have reported conflicting relationships among quality requirements [2, 3, 20-23].There are, however, different opinions as to the source of conflictive dependencies. Some authors have stated that conflict is inherent to a pair of quality requirements, while others emphasize that conflictive interactions depend on the software architecture and coding [22].

Several types of methods can be used to carry out a trade-off study, some of which depend on expert judgment, while others use semi-formal and formal models to compare alternatives [3]. The mapping study results in [9] reported that almost 50% (of 168 papers) deal with software architecture decision. Moreover, 25% of the papers address product quality and software process from a generic perspective. There are very few studies dealing with software coding and testing phases. The most common methods reported were the Analytical Hierarchy Process (AHP), model building, the Architecture Tradeoff Analysis Method (ATAM), algorithm-based and metric-based methods, expert opinion, Quality Function Deployment (QFD) and prototypes. At the analysis stage, the authors of the mapping study found specific techniques such as the Quality Performance (QUPER) model, prototyping and negotiation. With regard to the design stage, they reported additional techniques such as goals models, metrics, expert opinion and the automated construction of architecture alternatives. However, little empirical support is provided when software quality trade-offs are involved [9].

Software architecture trade-off methods have also been studied in order to understand the benefits and shortcomings of each one. Falessi et al. [24] compared decision-making techniques at the software design stage, taking into account the difficulties involved in using it. They found that there is no the best decision-making technique for the resolution of trade-offs in architecture design. In addition, Babar [25] proposed a framework with which to compare and evaluate various software architecture evaluation methods. Of the nine methods evaluated, only ATAM has the goal of analyzing trade-offs.
With regard to the development of decision-making processes based on ISO/IEC 12207 [11] or CMMI-Dev1.3 [10], we found that the Decision Analysis and Resolution (DAR) process area has been considered to define decision-making processes as regards the domains of both commercial-off-the-shelf (COTS) components and outsourcing companies. In the former category, Vantakavikran and Prompoon [26] described a process model with three layers, and they mapped each activity with DAR process area goals and practices. Phillips and Polen [27] described the Comparative Evaluation Process (CEP). They suggested set of criteria included the commonly evaluated characteristics (function, costs, maintainability and installation) and others that impact on management, architecture and strategic goals. They also considered contextual project factors and the credibility of data source. In the latter category, Hayshi [28], established a decision-making process with which to select outsourcing companies by considering the DAR process, and the AHP technique was used to prioritize criteria. This author also classified criteria as being either absolute or relative in order to reduce the number of alternatives when they did not meet the absolute criteria.

Finally, with regard to the harmonization approach, Pino et al. [13] reported that the Decision Management (DM) process from ISO/IEC 12207 has a partial relationship with practices from the DAR process area, but did not include the details of this comparison. Indeed, a current research line in harmonization is that of mapping models to discover the common practices among models [13, 29]. However, we wish to enrich this research area by using these mappings to build new processes.

3 Comparison between CMMI and ISO/IEC 12207 decision processes

Harmonization is an approach whose goals consist of deploying diverse quality improvement models in organizations, optimizing resources, and simultaneously obtaining the expected benefits of each model and achieving business goals [30]. In particular, we are interested in the identification of requirements for the decision-making process based on CMMI [10] and ISO12207 [11], in order to define a process whose goal is to support quality trade-off decisions. Requirements were extracted from models by carrying out a comparison between the decision processes of both models by adapting the techniques proposed in García-Mireles [12] and Pino et al. [13]. The activities involved in this comparison were:

1. Analyze models. The purpose of this is to understand the improvement models' goals, structure, and requirements. In this study, we describe the decision processes involved in the comparison.
2. Design mapping. The purpose of this is to set out a comparison procedure and to design mapping templates. We are interested in the details of the implementation of quality requirements. We then consider additional information such as sub-practices, notes and process outcomes.
3. Execute mapping. The purpose of this is to apply a comparison procedure in order to achieve mapping results.
4. Establish quality requirements. The purpose of this is to report requirements identified with regard to each model and to propose a solution that will allow them to be integrated into a process definition.

4.1 Analyze models

The Decision Analysis and Resolution (DAR) Process Area of CMMI-DEV1.3 belongs to the support category and appertains to the defined process level (maturity level 3). The process relies on a systematic evaluation process and established criteria to evaluate identified alternatives. Although it is recognized that uncertainty is one of the principal risks when making a decision, the model emphasizes a rationalistic approach to select, monitor and control either methods or selection criteria. The process also seeks new alternatives when the alternatives evaluated do not meet the stated requirements. The process area describes the specific goals, practices, sub-practices and exemplary of possible outcomes. There are also informative notes.

The Decision Management Process (DM), on the other hand, belongs to project processes within the main system context process group of the ISO/IEC 12207. The process analyzes project alternatives in order to then select one of them. The project must confirm that the preferred alternative resolves the issue that a request for a decision has identified. The process relies on the decision-making strategy to deal with decisions. The process specifies activities, tasks, and outcomes. There are also informative notes.

4.2 Design mapping

We are interested in identifying common actions and the differences among process elements in order to provide support when an organization wishes to define a decision process to support trade-off studies. We use a matrix in which we can see the relationships between the elements of both models, and include the work products (outcomes) used or produced in each practice or activity. The heading of Table 1 depicts the template elements.

4.3 Execute mapping

The results attained after comparing both models are presented in Table 1. We found that all the tasks and activities from the DM process are related to the DAR process practices. The processes' outcomes are aligned with DAR specific practices.

Table 1. Mapping between DAR y DM processes (Legend: SUB: sub-practices, SP: Specific practice; DP: Decision planning, DA: Decision analysis, DT: Decision tracking)

CMMI specific practices	ISO/IEC 12207:2008 activities and tasks	CMMI outcomes	ISO12207 outcomes
SP 1. Establish guidelines for decision analysis (100%)		Guidelines for when to apply a formal evaluation process	Decision-making strategy, applicable policies and procedures
SUB 1.	DP Task 1		
SUB 2.	6.3.3.3		
SP 2. Establish evaluation criteria (33%)		Documented evaluation criteria, rankings of criteria importance	Decision-making strategy
SUB 1.	DP Tasks 1, 2, 3		
SUB 2.	DA Task 1		
SUB 3, 4, 5,6			
SP 3. Identify alternative solutions (33%)		Documented evaluation criteria, identified alternatives	Alternative courses of action, decision-making strategy
SUB 1.			
SUB 2.			
SUB 3.	DP Task 3		
SP. 4. Select evaluation methods (33%)		Selected evaluation methods	Decision-making strategy
SUB 1.	DA Task 1		
SUB 2, 3			
SP 5. Evaluate alternative solutions (66%)		Documented evaluation criteria, rankings of criteria importance, identified alternatives, selected evaluation methods, evaluation results	Decision-making strategy, alternative courses of action, resolution, decision rationale and assumptions, (preferred courses of action)
SUB 1.	DA Task 2		
SUB 2.	DT Task 1		
SUB 3.	DT Task 1		
SUB 4, 5			
SUB 6.	DT Tasks 1, 2		
SP 6. Select solutions (50%)		Recommended solutions	Resolution, decision rationale and assumptions, (preferred courses of action)
SUB 1.			
SUB 2.	DT Task 2		

3.1 Establish quality requirements

The comparison results show that there is an overlapping between the DM and DAR processes. Indeed, the task of the DM process can be 100% addressed by DAR practices. However, the DM process covers only 52% of DAR practices. The main differences between the process outcomes are the terms used to name informational elements in both models. DM includes a decision-making strategy as a basis to capture all relevant data related to carrying out a decision-making process, while the DAR process provides more details as regards dealing with process outcomes. Since the DM process can be embedded in the DAR process, we use the latter as a basis to describe process quality requirements. Although we rely on the DAR specific practices, the proposal can help to understand the activities required to support trade-off studies.

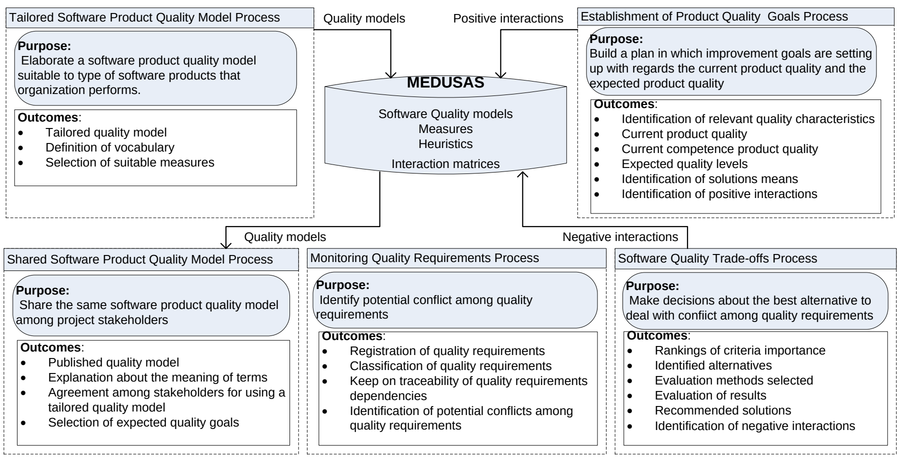
4 Framework to support software quality trade-offs

The research presented in this paper has been conducted in the context of the industrial project MEDUSAS (Improvement and Evaluation of Software Maintainability, Security, Usability and Design) whose goal is to build an ISO25000-based environment to support quality control and quality management. The project scope includes the assessment of both code and design models in order to determine the maintainability, security and usability of software products. The methodological component of MEDUSAS contains a quality assurance methodology in order to provide companies that require software quality control support with an independent assessment service. Moreover, the quality models were built in order to link terms, concepts, measures and heuristics to the aforementioned software quality characteristics.

One important issue that emerged once the quality models, measures, heuristics and checklist of security, maintainability and usability quality characteristics had been proposed was how to suitably manage the tradeoffs among them. This resulted in our proposal for a process framework to manage the interactions among quality requirements. This framework is composed of conceptual, methodological and technological elements. This paper focuses on the methodological component in which the processes required to make a trade-off decision are delineated. The set of processes described here are related to the MEDUSAS quality assurance methodology and support the monitoring and control of conflicts among quality attributes. Fig. 1 shows the principal processes in addition to a repository of models, methods and techniques used to perform trade-offs, and the dependences matrices among quality requirements.
The processes at the top of Fig. 1 correspond to the roles responsible for improving processes. These processes lead to the tailoring of a software quality model that is appropriate for the kind of software products that a company develops in which critical quality attributes are established and suitable measures are identified and evaluated. The establishment of a product quality goal process then takes place to diagnose the quality of both the company’s products and those of the competition in order to set a benchmark to permit the identification of future quality levels of critical attributes, and the mechanisms employed to meet those quality levels are documented.

The processes depicted at the bottom of Fig. 1 support conflict management when software engineers are developing a software product. The first attempts to achieve a common understanding as regards the software quality vocabulary. The second process seeks potential conflicts among quality requirements. The process identifies these by using the interaction matrices to check dependencies among quality requirements. If a conflict is detected, then the software quality trade-off process is performed.

As this paper addresses tradeoffs, we present a workflow of the activities in the generic process trade-off study process (Fig. 2). The activities correspond to DAR specific practices, but the identification of a trade-off situation is also added in order to highlight its relevance. The right-hand side of Fig. 2 shows the work products built throughout the process, while the left-hand side depicts the products built by the other framework processes, with the exception of negative interaction in which the problems among quality characteristics are documented in the MEDUSAS repository.



Tailored Software Product Quality Model Process
Purpose: Elaborate a software product quality model suitable to type of software products that organization performs.
Outcomes: Tailored quality model; Definition of vocabulary; Selection of suitable measures
Quality models
Positive interactions
MEDUSAS
Software Quality models
Measures
Heuristics
Interaction matrices
Establishment of Product Quality Goals Process
Purpose: Build a plan in which improvement goals are setting up with regards the current product quality and the expected product quality
Outcomes: Identification of relevant quality characteristics; Current product quality; Current competence product quality; Expected quality levels; Identification of solutions means; Identification of positive interactions
Quality models
Negative interactions
Shared Software Product Quality Model Process
Purpose: Share the same software product quality model among project stakeholders
Outcomes: Published quality model; Explanation about the meaning of terms; Agreement among stakeholders for using a tailored quality model; Selection of expected quality goals
Monitoring Quality Requirements Process
Purpose: Identify potential conflict among quality requirements
Outcomes: Registration of quality requirements; Classification of quality requirements; Keep on traceability of quality requirements dependencies; Identification of potential conflicts among quality requirements
Software Quality Trade-offs Process
Purpose: Make decisions about the best alternative to deal with confiict among quality requirements
Outcomes: Rankings of criteria importance; Identified alternatives; Evaluation methods selected; Evaluation of results; Recommended solutions; Identification of negative interactions

Fig. 1. Framework used to deal with software quality trade-offs

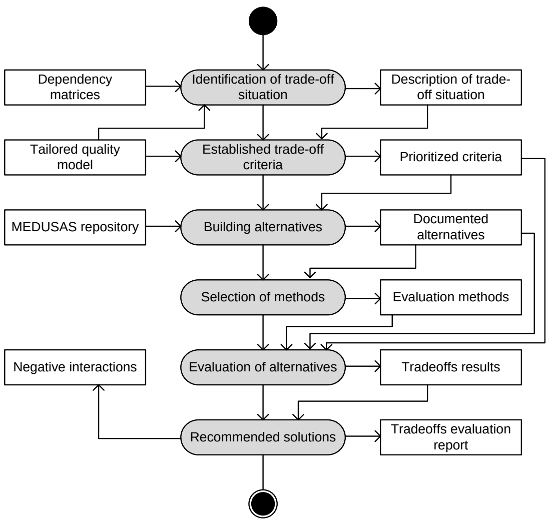


Fig. 2. Activities and products of software quality tradeoff process

Each activity is composed of several tasks. For instance, the establishment of trade-off criteria activity consists of the tasks described in Table 2. We also suggest some techniques that can be used to carry out these tasks. Maintainability, security and usability are the main quality characteristics that have been considered in MEDUSAS, and we have reviewed literature in order to identify conflictive interactions among them. We are currently surveying the companies that are working on this project in order to explore how they are dealing with conflicts among quality characteristics. An excerpt of the contents of this survey is shown in Annex A.

Table 2. Tasks from establishment of trade-off criteria activity

Tasks	Products	Methods or techniques
Select quality characteristics from tailored quality models as criteria	Input: tailored quality models, description of trade-off situation Output: selected quality characteristics	GQM in order to identify goals from relevant stakeholder
Add additional criteria for evaluation of alternatives	Output: additional criteria	Meetings or interviews to add other criteria, e.g. costs and utility
Identify compulsory criteria	Output: compulsory criteria	Interviews or meetings to identify the quality criteria that are compulsory to all alternatives
Determine criteria' priorities	Output: weight of each criterion	This could be determined using the AHP technique
Pilot criteria and ranking	Output: result of testing criteria and ranking	Apply criteria in an alternative to verify that criteria are useful. Verify that criteria are traceable to requirements, business objectives or other sources

5 Conclusions

If software development organizations are interested in improving their product quality, then they must consider how to deal with quality requirements, and particularly with any negative interactions among them. We have reviewed the main contributions from the quality requirement research area that are focused on the analysis and design stages of the software lifecycle. We have also enumerated some methods used to perform software quality trade-offs. Although this research area is still in the development phase, software organizations should improve the management of dependencies among quality requirements.

In order to support software organizations, we have proposed a framework to deal with issues regarding quality attribute interactions. This framework relies on requirements extracted from process improvement models and from issues to be considered when a project attempts to manage software product quality attributes. The software quality trade-off process, which is part of the framework, considers the requirements for decision-making processes found in CMMI and ISO/IEC 12207. We have used a harmonization technique to compare the processes in order to discover overlaps between DM and DAR. Since tasks from the DM process are covered by DAR process practices, we decided to take the latter as our primary source of requirements. We have also shown the main issues and recommendations that must be considered when a project deals with quality requirements.

The process framework addresses the tailoring of product quality models and the establishment of quality goals. At a project level, the processes attempt to ensure that stakeholders maintain an agreed meaning of quality attributes, monitor potential negative interactions among quality attributes, and finally, perform the software quality trade-off supported by rational selection of the alternatives that lead to a balance among quality attributes. The framework maintains the data regarding quality attribute interactions in order to assist with decision-making, and particularly trade-off processes. We are currently studying how companies identify and resolve conflicts among quality requirements, in the context of MEDUSAS project.

As future work, we are considering whether the proposed framework will be sufficiently flexible to deal with different quality standards. The conformance requirements could also be used to classify diverse trade-off methods. In order to attain the benefits of interaction management, it is necessary to propose a method with which to capture the knowledge concerning both positive and negative types of interactions. Traceability, another important issue, must be implemented in order to manage interactions and dependencies. It is also necessary to analyze the trade-off methods and understand how they can be linked to the proposed framework. We are currently exploring companies' awareness as regards interactions among quality requirements in order to refine our proposal.

References

1. Barbacci, M., M. Klein, T. Longstaff, and C. Weinstock. Quality Attributes (CMU/SEI-95-TR-021). 1995. Available from: http://www.sei.cmu.edu/library/abstracts/reports/95tr021.cfm
2. Boehm, B. and H. In: Identifying quality-requirement conflicts. IEEE Software. 13(2), 25-35 (1996)
3. Berander, P., et al. Software Quality Attributes and trade-offs. 2005. Available from: http://www.uio.no/studier/emner/matnat/ifi/INF5180/v10/undervisningsmateriale/reading-materials/p10/Software_quality_attributes.pdf
4. Paech, B. and D. Kerkow: Non-Functional Requirements Engineering - Quality is essential. In: B. Regnell, E. Kamsties, and V. Gervasi (eds.) 10th Anniversary International Workshop on Requirements Engineering: Foundation of Software Quality (REFSQ'04). pp. 237-250. (2004)
5. Chung, L. and J.C.S. Do Prado Leite: On non-functional requirements in software engineering. In: A. Borgida, et al. (eds.), pp. 363-379. Springer Berlin Heidelberg, (2009)
6. Robinson, W.N., S.D. Pawlowski, and V. Volkov: Requirements Interaction Management. ACM Computing Surveys. 35(2), 132-190 (2003)
7. Alexander, I.: Initial industrial experience of misuse cases in trade-off analysis. In Requirements Engineering, IEEE Joint International Conference on, pp. 61-68. (2002)

8. Chung, L. and B.A. Nixon: Dealing with non-functional requirements: three experimental studies of a process-oriented approach. In 17th international conference on Software engineering pp. 25-37. (1995)
9. Barney, S., et al.: Software quality trade-offs: A systematic map. Information and Software Technology. 54(7), 651-662 (2012)
10. CMMI, P.T. CMMI for Development, Version 1.3 (CMU/SEI-2010-TR-033). 2010. Available from: `http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm`
11. ISO, ISO/IEC 12207:2007. Systems and software engineering — Software life cycle processes. 2007
12. García-Mireles, G., M. Moraga, F. García, and M. Piattini: Towards the Harmonization of Process and Product Oriented Software Quality Approaches. In: D. Winkler, R. O'Connor, and R. Messnarz (eds.) Systems, Software and Services Process Improvement. 301, pp. 133-144. Springer Berlin Heidelberg. (2012)
13. Pino, F.J., et al.: Mapping software acquisition practices from ISO 12207 and CMMI. Journal of Software Maintenance and Evolution: Research and Practice. 22, 279-296 (2010)
14. Jedlitschka, A. and D. Pfahl: Towards Comprehensive Experience-Based Decision Support. In: T. Dingsøyr (eds.) Software Process Improvement. 3281, pp. 34-45. Springer Berlin Heidelberg. (2004)
15. Zannier, C., M. Chiasson, and F. Maurer: A model of design decision making based on empirical results of interviews with software designers. Information and Software Technology. 49(6), 637-653 (2007)
16. Aurum, A. and C. Wohlin: The fundamental nature of requirements engineering activities as a decision-making process. Information and Software Technology. 45(14), 945-954 (2003)
17. Ruhe, G.: Software Engineering Decision Support – A New Paradigm for Learning Software Organizations. In: S. Henninger and F. Maurer (eds.) Advances in Learning Software Organizations. 2640, pp. 104-113. Springer Berlin Heidelberg. (2003)
18. Ngo-The, A. and G. Ruhe: Decision Support in Requirements Engineering. In: A. Aurum and C. Wohlin (eds.) Engineering and Managing Software Requirements. pp. 267-286. Springer Berlin Heidelberg. (2005)
19. Van Lamsweerde, A., R. Darimont, and E. Letier: Managing conflicts in goal-driven requirements engineering. IEEE Transactions on Software Engineering. 24(11), 908-926 (1998)
20. Egyed, A. and P. Grünbacher: Identifying requirements conflicts and cooperation: How quality attributes and automated traceability can help. IEEE Software. 21(6), 50-58 (2004)
21. Zulzalil, H., A. Ghani, M. Selamat, and R. Mahmod: A Case Study to Identify Quality Attributes Relationships for Web-based Applications. IJCSNS International Journal of Computer Science and Network Security. 8(11), 215-220 (2008)
22. Henningsson, K. and C. Wohlin: Understanding the Relations between Software Quality Attributes - A Survey Approach(eds.) 12th International Conference for Software Quality. pp. 1-12. Ottawa, Canada, (2002)
23. Sadana, V. and X.F. Liu: Analysis of conflicts among non-functional requirements using integrated analysis of functional and non-functional requirements. In Computer Software and Applications Conference. COMPSAC 2007. 31st Annual International pp. 215-218. (2007)
24. Falessi, D., G. Cantone, R. Kazman, and P. Kruchten: Decision-making techniques for software architecture design: A comparative survey. ACM Computing Surveys. 43(4) (2011)

25. Babar, M.A., Z. Liming, and R. Jeffery: A framework for classifying and comparing software architecture evaluation methods. In Software Engineering Conference, 2004. Proceedings. 2004 Australian, pp. 309-318. (2004)
26. Vantakavikran, P. and N. Prompoon: Constructing a Process Model for Decision Analysis and Resolution on COTS Selection Issue of Capability Maturity Model Integration. In Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on, pp. 182-187. (2007)
27. Phillips, B.C. and S.M. Polen: Add decision analysis to your COTS selection process. CrossTalk. 21-25 (2002)
28. Hayshi, A.: Establish decision making process for selecting outsourcing company. In 21 International Conference on Softwre Engineering and Knowlegde Engineering, SEKE 2009., pp. 666-671. (2009)
29. Pardo, C., et al.: From chaos to the systematic harmonization of multiple reference models: A harmonization framework applied in two case studies. Journal of Systems and Software. 86(1), 125-143 (2013)
30. Siviy, J., P. Kirwan, J. Morley, and L. Marino. Maximizing your Process Improvement ROI through Harmonization. 2008. 1-16]. Available from: http://www.sei.cmu.edu/library/assets/whitepapers/multimodelExecutive_wp_harmonizationROI_032008_v1.pdf

Annex A. Partial questionnaire

Section 4 of this paper mentions a survey. The participants in this survey were requested to specify the quality attributes that they had addressed in the project. We then asked them the following, using a pairwise strategy

1. What kind of dependency did you observe in each pair of quality attributes? Positive, negative, independent or unidentified?
2. What rationale was used to determine this type of dependency?
3. In which life cycle stage was the dependency identified?
4. What means were used to meet this pair of quality characteristics?
5. What measures did you apply to verify the dependency?
6. What elements did you consider to evaluate the dependency as negative?
7. What procedures did you use to resolve the negative dependency?
8. What was the impact of negative dependencies on quality product requirements?
9. What was the impact of negative dependencies on software design, coding and testing?
10. Which participants were involved in the negative dependency identification and conflict resolution?