Risks and Risk Management in Software Architecture Evolution: an Industrial Survey

Odd Petter N. Slyngstad[1], Reidar Conradi[1], M. Ali Babar[2], Viktor Clerc[3], Hans van Vliet[3]
[1]*Department of Computer and Information Science (IDI), Norwegian University of Science and Technology (NTNU), {oslyngst, conradi} at idi.ntnu.no*
[2]*LERO, University of Limerick, Limerick, Ireland, malibaba at lero.ie*
[3]*Department of Computer Science, Vrije Universiteit, Amsterdam, the Netherlands, {viktor, hans} at cs.vu.nl*

Abstract

The effort that has been made to study risk management in the context of software architecture and its evolution, has so far focused on output from structured evaluations. However, earlier research shows that formal, structured evaluation is not commonly used in industry. We have performed a survey among software architects, in order to capture a more complete picture of the risk and management issues in software architecture evolution. Our survey is specifically about their current knowledge of actual challenges they have anticipated and experienced, as well as strategies they have employed in response. We received completely filled questionnaires from 82 respondents out of a total distribution of 511 architects from the software industry in Norway. While many of the risks we have identified can be aligned with results from earlier studies, we have also identified several risks which appear not to fit these risk categories. Additionally, we found a direct link to business risks, as well as a relatively low level of awareness that lack of software architecture evaluation represents a potential risk.

Keywords: *software architecture, software evolution, risk management, software architecture evaluation*

1. Introduction

Proper management of software architecture is one of the most important factors towards successful development and evolution of component-based software systems. The architecture is a core part of a software system [1], and obtaining and disseminating information about relevant risks is therefore important. By software architecture evolution, we mean accumulated changes to the structure(s) of a system of software elements, their external properties and mutual relationships [1]. Some effort has been made towards identifying and understanding the risks and corresponding strategies involved in managing software architecture evolution (but based on outputs from structured architecture evaluations – which are not commonly used in industry [7]).

We expect that practicing IT-professionals who work with architecture on a daily basis will encounter architectural evolution risks on a regular basis. Therefore, this investigation targets software professionals in the IT-industry with significant knowledge and experience about software architecture design and evolution. The systems we investigate are within Component-Based Software Engineering (CBSE – including internal, Components Off-The-Shelf (COTS) and Open Source components (OSS) development), and whose architectures have evolved during their lifetimes. We have also targeted small and medium sized enterprises (SMEs).

Based on our initial identification of risks and management strategies [19], we have carried out an industrial survey of architectural risks and mitigation strategies in software evolution. In this investigation, we have identified additional architectural risks and associated risk management strategies related to software evolution in the IT-industry, in a three-part adapted operational matrix (Tables 1, 2, 3). This allows easy lookup of strategies and related outcome profiles as applied to the most influential risks we identified. The most influential risks were regarding poor clustering of functionality and insufficient stakeholder communication. While many of the risks we have identified fit with results from earlier studies [22][23], we have also identified several risks which appear not to fit risk these risk categories. We found a low level of awareness that lack of architecture evaluation

1530-1362/08 $25.00 © 2008 IEEE
DOI 10.1109/APSEC.2008.70

represents a potential problem, as well as a direct link to business risks.

The remainder of this paper is organized as follows: Section 2 holds background and related work. The research method is presented in Section 3. Section 4 contains information on our data collection and analysis, and the results of our study stand in Section 5. Discussion and Threats to Validity are contained in Section 6, and conclusion and future work stand in section 7.

2. Background and related work

Risks are challenges that can have negative influences on a project unless they are handled properly. Risk management involves methods to handle risks that may occur during a software development project. Boehm [8] details a **risk management framework** which includes two main steps: risk assessment (i.e. risk identification, analysis and prioritization) and risk control (i.e. planning, resolution and monitoring). Technological, process and organizational issues are also important collective facets of software risk mitigation (i.e. proper handling of occurring problems to minimize their consequences) [11][12][13].

A definition of **Software Architecture** can be found in [1, page 3]: *the structure or structures of a system, which comprise software elements, the externally visible properties of those elements, and the relationships between those elements.* Defining and maintaining a software architecture properly is key to success with development and evolution of non-trivial systems, such as within CBSE [2]. Benefits to project organizational structure can also be seen. Furthermore, a lack of attention to software architecture often has negative consequences reaching beyond the architecture itself, for example with respect to inter-personal communication, unnecessary redundancies and decision making in the project [1].

Software evolution can be defined as: part of maintenance [5], a software lifecycle step [4], the dynamic behavior of software systems through lifelong maintenance and enhancements [3], and the enhancements and improvements regarding functionality and performance made between releases [9]. We define software evolution (updated from [19]) as follows: *the systematic and dynamic updating of a component (source code) or other artifact to a) accommodate new, altered or removed functionality or b) enhance the reliability/availability (i.e. fewer failures), performance, or other quality attribute(s) of such an artifact between different releases.*

Ropponen and Lyytinen performed a survey regarding risks in software development and how these risks can be handled [6]. They identified the following six categories of software risk: 1) scheduling and timing risks, 2) functionality risks, 3) subcontracting risks, 4) requirements management, 5) resource usage and performance risks, and 6) personnel management risks. Their results also reveal that all of the risk categories were affected by environmental factors.

Bass et al. [22] and O'Connell [23] used results from ATAM evaluations to reveal and analyze risk themes specifically towards software architecture. These studies utilize ATAM outputs from organizations where such evaluation is established as a practice, but do not comment on how common such formal evaluation methods are in industry. A related study [7] on architecture evaluation shows that practices range from completely ad-hoc to formally structured, from qualitative to quantitative. Additional risks and management strategies could therefore escape discovery when only investigating structured analysis reports. Another key point is that most of the needed architectural information, save perhaps the actual architectural design, commonly is not explicitly documented [21]. It should also be noted that neither of the studies [22][23] deal explicitly with evolution of software architecture. In contrast, our investigation incorporates risks explicitly identified in planning and experienced during the evolution of software architecture, based on input directly from software architects.

3. Research method

Motivation: Our initial observation [19] was that risk management studies usually identify risks on the general project level [6][8][10]. On the other hand, software architecture studies commonly focus on the design, implementation and maintenance of the architecture, i.e. as a software artifact. There has been some effort to study the two in combination [19], i.e. risk management of software architecture activities, but based on outputs from structured architecture evaluations [22][23]. We therefore decided to perform a more in-depth investigation to further identify and understand actual risks and associated risk management strategies in relation to software architecture evolution, as they are identified, experienced and employed in industry. We study industrial risk identification, analysis and prioritization (**RQ1**), as well as risk planning and resolution (**RQ2**) [8][19].

Our Research Questions (background in a review of research literature, being adapted as follows)**:**

RQ1: What are the relevant architectural evolution risks, i.e. what risks induced on the software architecture through software evolution? Boehm's first step [8] includes risk identification, analysis, and prioritization. We are focusing on the state-of-the-practice in risk awareness, i.e. we wish to gain insight regarding which risks software architects deem important with respect to an evolving architecture. In our prestudy [19] we saw that challenges (risks) were handled on a case-by-case basis, whether known before project start, or experienced during projects.

RQ2: How can these risks best be mitigated: how successful were the relevant risk management strategies? Boehm's second step [8] includes risk control, focusing on problem mitigation; i.e. proper handling of occurring problems to minimize their impact. Our goal is again to obtain an overview of state-of-the-practice. Furthermore, we wish to suggest possible improvements through enabling a systematic approach to managing architectural risks in software evolution. Hence, it is important that we obtain information regarding positive and negative aspects of applied risk management strategies and their outcomes. As mentioned, our prestudy [19] indicated that risk mitigation is performed in an ad-hoc manner. Most of the respondents also reported not having a documented or defined risk management process to deal systematically with risk aspects when altering the architecture.

Our questionnaire: Building on the questionnaire from our prestudy [19], we have designed a revised questionnaire totalling 23 questions. Questions Q1-Q5 and Q8 are closely related to those in our prestudy. Furthermore, questions Q6-Q23 were adapted and changed from an earlier empirical study aimed at identifying the factors that can influence software architecture evaluation practices [7]. Some of the questions are semi-open questions, i.e. the answer categories are indicated, but the actual answer is free text. The remaining questions are mainly closed, although some have alternatives such as "other, please describe:" to allow filling in additional answers not covered by the given alternatives for a particular question. In addition, respondent information regarding level of experience, education, number of years in the IT-industry, position, size of company etc. was collected. We have tried out the questionnaire on four of our colleagues to ensure the quality of the questions and that they were easy to understand. This caused 7 out of 23 questions to be refined.

In this article, we are focussing on identification, analysis and prioritization, as well as planning and resolution, of risks. We therefore discuss questions Q1.1 and Q1.2 of the questionnaire separately in this article, as both of them contribute to both RQ1 and RQ2. These questions read "Challenges **Q1.1: identified in planning, Q1.2: experienced during, the maintenance/evolution** related to the software architecture (indicate influence on the architecture, also indicate strategies, and their outcome)?"

In questions Q1.1 and Q1.2, we used a 5-point ordinal Likert scale to rank risk Influence with value range Very High (VH) = 5, High (H) = 4, Medium (M) = 3, Low (L) = 2 and Very Low (VL) = 1. Similarly, strategy Outcome was ranked Not at all = 1, Somewhat = 2, Medium = 3, Mostly = 4, and Completely = 5 successful. The rank 0 was used to indicate "Don't know" on both scales.

Context: Our research in this investigation is on software development projects with the following two major characteristics – use of CBSE (including development with internal, COTS and OSS components), and incurred changes in the software architecture during their lifetime. This implies that the investigated projects have delivered their first production release. That is, they can be considered to be in the "post-development" phase, i.e. undergoing maintenance/evolution. The survey respondents were all from the IT-industry in Norway.

4. Data collection and analysis

This questionnaire-based survey was performed using a variant of snowball sampling, a technique described in [18], where key practitioners serve as contact points towards the organizations involved. The contact points are then sent the questionnaire, and forward it on to other potential respondents. The contact points can also report the total number of respondents from each organization and function as a temporary checkpoint for the number of completed questionnaires. This type of sampling is close to convenience sampling in that the contact persons are known during the execution of the survey. To ease the workload and streamline the data collection and validation process, we enabled a web-interface to make the questionnaire available to the respondents online.

To ensure previous knowledge and experience working with software architecture, we required at least 2 years of professional experience (the lowest level seen in our prestudy – respondents at this level were still able to give relevant and valuable answers). The questionnaire took about 30 minutes to fill in completely. In total we were able to reach 63 small and medium sized software companies (all less than 100 employees), with 511 potential respondents. We received 82 complete answers out of 511 total contacted (i.e. a response rate of 16 %). Furthermore, the mean project size was 7.

We analyzed the data as follows: The data on risks and strategies were divided into distinct parts and each piece coded according to risk or strategy theme(s). As an example [19], for risks this was specified as {risk condition – what may go wrong, risk consequence(s)}. For example, "requirements from earlier versions still in effect affected architecture design negatively" was coded as {earlier version requirements, negative for architecture design}. The coded pieces were then examined for commonalities and differences, combining related information pieces. For example, for risks, {earlier version requirements, negative for architecture design} and {required same functionality as before, negative for planning} were grouped as {required backward compatibility, negative for architecture maintenance/evolution planning and design}. These groups were then compared to the risks and strategies we discovered from our prestudy to check for overlaps and similarities.

We ran all the answer records through this procedure. The results were further checked to ensure reliability. This is similar to the constant comparison method described in [16]. We retained the risk classification scheme used in our prestudy [19] for the three categories technical, process and organizational. We maintain that risk management is not merely a technical issue; rather, it covers all three categories [11][12][13].

5. Results

In presenting the results, we have divided the risks into the three categories mentioned above. The results are presented in three tables which together constitutes an adapted operational matrix. In this context they enable lookup of strategies and their outcome profiles as applied to the most influential risks we identified. The outcome is presented as a set of the number of instances each rating was given by the respondents, i.e. Outcome rating = Number of {"Not at all", "Somewhat", "Medium", "Mostly", and "Completely"} successful instances. Though it was possible to enter "Don't Know" as a rating, none of the respondents did so. In presenting the results, the strategy with the highest number of "Completely" (or "Mostly" if no instances were rated "Completely") successful responses is taken as the most successful. The strategy applied by the most respondents is taken as the most frequent strategy. Finally, "*" indicates that this risk was also identified among the most influential in our prestudy [19].

Technical risks: From Table 1, we can see that Technical Strategy (TS) 1 (Table 1) was the overall most successful strategy, and also the most frequent one, applied in planning. During maintenance/evolution, TS 7 (Table 1) was the most successful (and most frequent) strategy applied towards technical risks.

Table 1. Most influential (Risk Influence VH > 1) technical risks (TRs), and strategies

Technical (identified in planning), ID: Risk	**Risk Influence**	**ID:Strategy:Outcome rating** = Number of {"Not at all", "Somewhat", "Medium", "Mostly", and "Completely"} successful instances.		
TR 1: Poor clustering of functionality affected performance negatively *	VH: 7, H: 23	**TS 1**	Refactoring of the architecture	{0, 8, 2, 5, 3}
		TS 2	Redesign within constraints	{0, 0, 1, 4, 0}
		TS 3	Design with high focus on modifiability	{0, 1, 2, 6, 1}
		TS 4	Finalize modifiability design considerations early	{0, 1, 0, 0, 0}
TR 2: Requirements from other system(s) affected performance negatively	VH: 5, H: 10	**TS 2**	Redesign within constraints	{0, 1, 4, 4, 0}
		TS 5	Employ separate agents for external communication, protocol for information sharing	{0, 1, 2, 2, 0}
		TS 3	Design with high focus on modifiability	{0, 1, 4, 3, 0}
TR 3: Undefined variation points in requirements affected performance negatively, caused increased focus on modifiability	VH: 3, H: 10	**TS 3**	Design with high focus on modifiability	{0, 0, 3, 5, 1}
		TS 4	Finalize modifiability design considerations early	{0, 3, 2, 3, 0}
TR4: Extensive focus on streamlining of the architecture affected modifiability negatively	VH: 2, H: 10	**TS 3**	Design with high focus on modifiability	{0, 0, 3, 3, 1}
		TS 4	Finalize modifiability design considerations early	{0, 2, 3, 3, 0}
TR 5: Architectural mismatch caused redesign of part of the architecture	VH: 2, H: 2	**TS 1**	Refactoring of the architecture	{0, 1, 1, 0, 0}
		TS 3	Design with high focus on modifiability	{0, 0, 0, 1, 0}
		TS 4	Finalize modifiability design considerations early	{0, 0, 1, 0, 0}
Experienced during				
TR 6: Increased focus on modifiability contributed negatively towards system performance *	VH: 6, H: 10	**TS 6**	Implementation of changes towards improved modifiability	{0, 0, 2, 1, 0}
		TS 7	Minor implementation changes	{0, 1, 6, 7, 0}
TR 7: Poor original core design prolonged the duration of the maintenance/ evolution cycle *	VH: 3, H: 11	**TS 6**	Implementation of changes towards improved modifiability	{0, 0, 3, 4, 0}
		TS 8	Informal review of the architecture	{0, 0, 3, 3, 0}
		TS 7	Minor implementation changes	{0, 0, 1, 2, 0}
		TS 1	Refactoring the architecture	{0, 0, 3, 0, 0}
TR 8: Varying release cycles for COTS/OSS components made it difficult to implement required changes *	VH: 2, H: 16	**TS 9**	Use own development as potential backup solution	{0, 4, 5, 8, 0}
		TS 10	Implement extra architecture add-ons	{0, 1, 2, 0, 0}

Process risks: The results from Table 2 show that 7 out of 12 Process Strategies (PS) applied in planning were rated Completely (i.e. Outcome = 5) successful in at least one instance. For strategies applied during the

maintenance/evolution, only 1 out of 12 (PS 17) strategies was rated Completely successful in one instance. Furthermore, PS 1 was the most successful and most frequent strategy used.

Table 2. Most influential (Risk Influence VH > 1) process risks (PRs), and strategies

Process (identified in planning), ID: Risk	**Risk Influence**	**ID:Strategy:Outcome rating** = Number of {"Not at all", "Somewhat", "Medium", "Mostly", and "Completely"} successful instances.		
PR 1: Lack of architecture documentation required more effort to be spent on planning during maintenance/evolution *	VH: 6, H: 25	**PS 1**	Recover needed architecture documentation using current architecture design and other artefacts as a basis	{0, 3, 2, 5, 1}
		PS 2	Thorough planning before implementing maintenance/evolution changes	{0, 1, 8, 7, 1}
		PS 3	Recover architecture evaluation artefacts where needed	{0, 0, 4, 2, 0}
		PS 4	Alter process to capture important architecture details	{0, 1, 3, 3, 0}
		PS 5	Explicit training on architecture documentation	{0, 0, 1, 3, 0}
PR 2: Lack of architecture evaluation contributed to discovering potential problems later in planning of maintenance/evolution	VH: 5 , H: 13	**PS 1**	Recover needed architecture documentation using current architecture design and other artefacts as a basis	{0, 0, 3, 4, 1}
		PS 3	Recover architecture evaluation artefacts where needed	{0, 1, 2, 4, 0}
		PS 4	Alter process to capture important architecture details	{0, 0, 2, 3, 0}
PR 3: Lack of business context analysis affected stakeholder relationships negatively	VH: 4 , H: 13	**PS 6**	Integrate business context in planning of the maintenance/evolution	{0, 2, 5, 3, 1}
		PS 7	Include business context informally	{0, 1, 1, 4, 0}
PR 4: Insufficient requirements negotiation postponed important architecture decisions	VH: 4, H: 9	**PS 8**	Negotiate requirements early	{0, 0, 2, 2, 1}
		PS 9	More explicit communication	{0, 2, 3, 0, 0}
		PS 10	Allow additional time for communication and feedback	{0, 1, 1, 3, 0}
Experienced during				
PR 5: Insufficient stakeholder communication contributed to insufficient requirements negotiation and affected implementation of new/changed architectural requirements negatively	VH: 7, H: 13	**PS 13**	Extra communication effort	{0, 1, 7, 3, 0}
		PS 14	Postpone some requirements to next maintenance/evolution cycle	{0, 0, 1, 2, 0}
		PS 15	Arrange plenary meetings for all stakeholders	{0, 0, 3, 4, 0}
		PS 16	Negotiate project extension	{0, 1, 2, 2, 0}
PR 6: Poor integration of architecture changes into implementation process affected implementation process and the architecture design negatively *	VH: 2, H: 20	**PS 17**	Overlay architecture change process onto implementation of maintenance/evolution	{0, 0, 4, 7, 1}
		PS 18	Integrate architecture considerations into implementation process	{0, 1, 9, 2, 0}

Organizational risks: Among the Organizational Strategies (OS) used in response to the most influential organizational risks (Table 3) identified in planning, 2 out of 9 (OS 2 and OS 4) were Completely successful in one instance. The highest rating for the strategies employed towards organizational risks experienced during the maintenance/ evolution was Mostly successful. Furthermore, OS 5 was the overall most frequent strategy applied towards the organizational risks identified.

Table 3. Most influential (Risk Influence VH > 1) organizational risks (ORs), and strategies

Organization (identified in planning), ID: Risk	**Risk Influence**	**ID:Strategy:Outcome rating** = Number of {"Not at all", "Somewhat", "Medium", "Mostly", and "Completely"} successful instances.		
OR 1: Not allowed to change OSS as decision mandate external to architecture team, affecting performance and modifiability negatively *	VH: 6, H: 22	**OS 1**	Frequent, interactive, scheduled meetings to keep up to date	{0, 1, 4, 5, 0}
		OS 2	Involve all "layers" of customer organization as stakeholders, allow extra time for proper communication	{0, 0, 1, 0, 0}
		OS 3	Ensure compliance with external mandate holder	{0, 0, 4, 1, 0}
		OS 4	Involve mandate holder early as stakeholder in planning	{0, 2, 4, 9, 1}
OR 2: Separate architecture team per maintenance/evolution cycle basis contributed to loss of and insufficient knowledge about the existing architectural design *	VH: 4, H: 29	**OS 5**	Dedicate personnel to "retrieve" architecture knowledge	{0, 2, 11, 6, 0}
		OS 6	Increased focus on proper documentation, to allow bringing new personnel up to speed quickly	{0, 1, 8, 6, 0}
OR 3: Cooperative maintenance/evolution with architects from customer org. required extra training and communication efforts *	VH: 3, H: 10	**OS 1**	Frequent, interactive, scheduled meetings to keep up to date	{0, 0, 1, 1, 0}
		OS 7	Perform maintenance/evolution incrementally	{0, 0, 2, 0, 0}
		OS 8	Allott extra time for proper communication with all stakeholders	{0, 0, 1, 0, 0}
		OS 9	Include other project's architects in planning, implementation	{0, 1, 4, 5 0}
OR 4: Lack of clear point of contact from customer organization contributed to inconsistencies in communication of the architecture and requirements *	VH: 2, H: 27	**OS 5**	Dedicate personnel to "retrieve" architecture knowledge	{0, 0, 1, 0, 0}
		OS 1	Frequent, interactive, scheduled meetings to keep up to date	{0, 0, 4, 5, 0}
		OS 2	Involve all "layers" of customer organization as stakeholders, allow extra time for proper communication	{0, 3, 6, 4, 1}
		OS 6	Increased focus on proper documentation, to allow bringing new personnel up to speed quickly	{0, 1, 5, 6, 0}
Experienced during				
OR 5: Prior architecture maintenance/evolution pushed to other projects due to lack of personnel influenced knowledge on the architecture negatively *	VH: 3, H: 11	**OS 10**	Regain architecture details from remaining upper management personnel	{0, 0, 2, 1, 0}
		OS 11	Keep architecture documentation centralized	{0, 0, 5, 8, 0}
OR 2: Separate architecture team per maintenance/evolution cycle contributed to loss of and insufficient knowledge about the existing architectural design *	VH: 2, H: 13	**OS 10**	Regain architecture details from remaining upper management personnel	{0, 2, 6, 6, 0}
		OS 11	Keep architecture documentation centralized	{0, 0, 0, 1, 0}
		OS 12	Set up standard procedure for distribution of architecture documentation and knowledge	{0, 2, 0, 0, 0}

Furthermore, our results show that the overall most influential risk was TR 1: "Poor clustering of functionality affected performance negatively". The corresponding most successful strategies were TS 1: "Refactoring of the architecture" and TS 3: "Design with high focus on modifiability". The second most influential risk was PR 5: "Insufficient stakeholder communication contributed to insufficient requirements negotiation and affected the implementation of new/changed architectural requirements negatively", with PS 15: "Extra communication effort" and PS 17: "Arrange plenary meetings for all stakeholders" as the corresponding most successful strategies. The outcome rating mode was "High" for Technical, "Medium","High" for Process, and "Medium" for Organizational strategies applied towards these most influential risks (Tables 1, 2, 3). The median outcome rating was "Medium" for all three categories.

6. Discussion

Comparison with related work: Table 4 shows the relation between risk categories identified by Ropponen et al. [6] (general software development risk categories) and Bass et al. [22] (architectural risk categories). The relations shown indicate the industrial relevance of the risks identified in our investigation.

Table 4. Summary of comparison to related work

Ropponen et al. [6]	**Technical risks (TR)**	**Process risks (PR)**	**Organizational risks (OR)**
Requirements risks:	TR 2, TR 3, TR 6,	PR 4	
Architecture Team risks:			OR 2
Stakeholder risks (from the subcontractor viewpoint):		PR 5	OR 3, OR 4
Bass et al. [21]			
Quality Attribute risk:	TR 6		
Integration risks:	TR 5, TR 8	PR 6	OR 1
Requirements risks:	TR 3	PR 4	
Documentation risks:		PR 1	
Process and Tools risks:		PR 2	
Allocation risks:	TR 1, TR 7, TR 8		
Coordination risks:		PR 5	OR 1, OR 2, OR 3, OR 4

In summary, three of the architectural risks we have identified do not fit the categories in related work [22][23]. We have focused specifically on architectural risks as {risk, consequence} (see Section 4), while earlier studies focused on aggregating categories of risks. Furthermore, we have identified relevant strategies towards mitigating the identified risks in software architecture evolution (Tables 1, 2, 3).

Observations on key architectural risks and promising risk management: The three-part adapted operational matrix in Tables 1, 2 and 3 enables lookup of strategies and related outcome profiles as applied to the most influential risks we identified, for both practitioners and researchers. Our aim is that researchers will use this matrix to build on in further investigations of risks and strategies in software architecture evolution. Practitioners can use this matrix to gain information on relevant strategies for use in response to risks they encounter.

Comparing our results with our prestudy [19], TR 1 (overall most influential risk) is identical to the second overall most influential risk. PR 5 (second overall most influential risk) is also closely related to the risk identified as overall most influential in that study. Furthermore, 4 out of 8 Technical risks, 2 out of 6 Process risks and 4 out of 5 Organizational risks identified in this investigation as most influential were also identified in our prestudy [19]. The larger number of identified risks appears in planning, as opposed to being encountered later during the maintenance / evolution, which was the case in our prestudy [19]. Furthermore, PR 3 shows a more **direct link to Business Risks [24]** (i.e. risks which influence software system viability) than was discovered in [19]. This comes in addition to the implicit circular feedback influences on and from e.g. normal cost and schedule monitoring.

Defined and documented architecture evaluation enables architects to e.g. discover design errors and conflicting requirements early in the process, potentially saving a project from more significant problems later. In this investigation, we find risks that mirror this concern, such as PR 2. Nevertheless, only 18 out of 82 respondents indicate this risk's influence as "Very High" or "High". While there is **a relatively low level of awareness that lack of architecture evaluation represents a potential risk**, the corresponding mitigation strategies we identified (PS 1, PS 3, PS 4) merely entail recovering the missing evaluation output. However, there is some evidence in other research that internal architecture evaluation is frequently performed by experts, and works because of their high level of competence and experience [25].

The median strategy outcome rating in all three categories (technical, process, organizational) was "Medium", indicating that there is still need for improvement in mitigating risks. While a large number of the identified Technical strategies (TS) focus on developing the specified system, the majority of the identified Process strategies (PS) involve recovering needed architectural documentation or other details. Furthermore, the majority of the Organizational strategies include efforts towards better communication with e.g. stakeholders.

The focus of architects' mitigation efforts are hence on recovering needed architecture details and improving communication while producing the system according to specification. Effort should therefore be made towards improving regular documentation and evaluation of the architecture, integrated with the maintenance / evolution process. Proper training of both architects and organizational management are means to achieve these improvements.

Threats to Validity in our investigation are here discussed based on Wohlin et al. [15]:

Construct Validity: The research questions have a firm basis in the research literature. The actual questionnaire questions have been mapped directly to the research questions. The survey questionnaire has been further pre-tested through four colleagues to ensure its quality. The questionnaire questions that were not adopted from a previous study [7] (see Section 3) were initially investigated in our prestudy [19]. Furthermore, all terminology used in the questionnaire is explained at the start of the questionnaire to provide clear definitions and avoid misinterpretations.

External Validity: This survey has been conducted by using non-probabilistic snowball-like sampling [18]. It is very difficult to achieve a random sample in surveys within the software engineering field, due to the lack of good demographic information regarding the populations we are interested in, though an example of stratified-random sampling of projects has been described in the research literature [20]. Furthermore, we ensured the total sampling frame (511 professionals) had relevant background and experience in software architecture. All the respondents are nevertheless from the Norwegian IT-industry, an issue which remains a limitation.

Internal Validity: All the respondents had relevant knowledge of and experience with industrial software development. They have also expressed an interest in the survey, so we think that they have answered the survey questions to the best of their ability by relying on their own experiences, skills and knowledge of software architecture. We were also available via email during the survey to clarify any ambiguities in the questions or the accompanying terminology definitions.

Conclusion Validity: This is a qualitative study, and we have used non-probabilistic snowball-like sampling. The number of respondents is 82 (out of 511), and were all from small and medium companies (all less than 100 employees), with a mean project size of 7 person-years.

7. Conclusion and future work

Our survey on risks and risk management regarding software architecture evolution has involved 82 respondents from the software industry. Through this survey on state-of-practice, we have identified real, industrial, architectural risks and corresponding management strategies employed in response.

We have developed a three-part adapted operational matrix (Tables 1, 2, 3) for risks and corresponding risk management strategies in software architecture evolution, based on responses from our survey respondents. Table 5 shows a summary of our findings.

Table 5. Summary of findings

Most influential risks	Corresponding most successful strategies
1. "Poor clustering of functionality affected performance negatively"	"Refactoring of the architecture" and "Design with high focus on modifiability"
2. "Insufficient stakeholder communication contributed to insufficient requirements negotiation and affected the implementation of new/changed architectural requirements negatively"	"Extra communication effort" and "Arrange plenary meetings for all stakeholders"
Additional findings: • Direct link to Business risks. • Relatively low level of awareness that lack of architecture evaluation represents a potential risk.	

Future work involves expanding our survey base to other countries (e.g. Netherlands). We also want to couple our findings here with an investigation of code-level and artefact data related to architecture evolution, in order to move towards a framework for better handling of these issues. Finally, a more thorough analysis of software architecture evaluation methods, processes and issues is planned.

8. Acknowledgements

This research is performed in cooperation between NTNU, Vrije Universiteit Amsterdam, and the Lero center in Limerick. The study was performed in the SEVO (Software EVOlution in component-based software engineering) project (SEVO, 2007), a Norwegian R&D project in 2004-2008 with contract number 159916/V30. It also falls under the Griffin project umbrella at Vrije Universiteit Amsterdam. Ali Babar is supported by Science Foundation Ireland's grant 03/CE2/I303_1.

9. References

[1] L. Bass, P. Clements, R. Kazman, *Software Architecture in Practice,* Second Edition, Addison-Wesley, 2004.

[2] L. Bass, C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord, K. Wallnau, *Volume I: Market Assessment of Component-based Software Engineering* in SEI Technical Report number CMU/SEI-2001-TN-007, 2001.

[3] L. A. Belady and M. M. Lehman; *A model of a Large Program Development,* IBM Systems Journal, 15(1):225-252, 1976.

[4] K. H. Bennett and V. Rajlich; *Software Maintenance and Evolution: A Roadmap,* ICSE'2000 – Future of Software Engineering, Limerick, Ireland, pp. 73-87, 2000.

[5] I. Sommerville; *Software Engineering,* Seventh Edition, Addison-Wesley, 728 p., 2004.

[6] J. Ropponen and K. Lyytinen, *Components of Software Development Risk: How to Address Them? A Project Manager Survey,* IEEE Trans. Sw. Engr., 26(2):98-112, Feb. 2000.

[7] M. Ali Babar, L. Bass, I. Gorton, *Factors Influencing Industrial Practices of Software Architecture Evaluation: An Empirical Investigation*, Proceedings of QoSA 2007, Springer LNCS 4880, pp. 90-109, Medford, Massachusetts, USA, July 12-13, 2007.

[8] B. W. Boehm, *Software Risk management: Principles and Practices,* IEEE Software, 8(1), 32-41, January 1991.

[9] P. Mohagheghi and R. Conradi, *An Empirical Study of Software Change: Origin, Acceptance Rate, and Functionality vs. Quality Attributes*, ISESE 2004, Redondo Beach (Los Angeles), USA, 19-20 Aug. 2004.

[10] M. Keil, P. E. Kule, K. Lyytinen and R. C. Schmidt, *A Framework for Identifying Softare Project Risks,* Communications of the ACM, 4(11), 76-83, November 1998.

[11] B. W. Boehm, *A Spiral Model of Software Development and Enhancement,* IEEE Computer, 21(5), 61-72, May 1988.

[12] A. Gemmer, *Risk Management: Moving Beyond Process,* IEEE Computer, 30(5), 33-41, May 1997.

[13] H. Hecht, *Systems Reliability and Failure Prevention,* Artech House Publishers, 2004.

[14] V. Clerc, P. Lago, H. van Vliet, *The Architect's Mindset*, Proceedings of QoSA 2007, Springer LNCS 4880, pp. 231-249, Medford, Massachusetts, USA, July 12-13, 2007.

[15] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Wesslén, *Experimentation in Software Engineering – An Introduction,* Kluwer Academic Publishers, 2002.

[16] A. L. Strauss and J. M. Corbin, *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*, Sage Inc., 1998.

[17] T. C. Lethbridge, S. E. Sim, and J. Singer, *Studying Software Engineers: Data Collection Techniques for Software Field Studies*, Empirical Software Engineering, 10(3):311-341, July 2005.

[18] B. Kitchenham and S. L. Pfleeger, *Principles of Survey Research, Parts 1-6,* ACM Software Engineering Notes, 2001–2002.

[19] O. P. N. Slyngstad, J. Li, R. Conradi, M. Ali Babar, *Identifying and Understanding Architectural Risks in Software Evolution: An Empirical Study,* Accepted at Profes '08, 15p, forthcoming in a Springer LNCS, June 2008.

[20] R. Conradi, J. Li, O. P. N. Slyngstad, V. B. Kampenes, C. Bunse, M. Morisio and M. Torchiano, *Reflections on conducting an international survey of Software Engineering,* in J. Verner and G. H. Travassos (Eds.): Proc. Int'l Symposium on Empirical Software Engineering (ISESE'05), pp. 214-223, Noosa Heads (Brisbane), Australia, 17-18 Nov. 2005, IEEE Computer Society, 2006.

[21] P. Kruchten, P. Lago, H. van Vliet, Timo Wolf, *Building up and Exploiting Architectural Knowledge*, Proc. WICSA 2005, pp. 291-292, IEEE Computer Society 2006.

[22] L. Bass, R. Nord, W. Wood, D. Zubrow, *Risk Themes Discovered Through Architecture Evaluations,* Proc. WICSA 2007, pp. 1-10, IEEE Computer Society, 2007.

[23] D. O'Connell, *Boeing's Experiences using the SEI ATAM® and QAW Processes*, April, 2006, http://www.sei.cmu.edu/architecture/saturn/2006/OConnell.pdf

[24] D. G. Messerschmitt and C. Szyperski, *Marketplace Issues in Software Planning and Design*, *IEEE Software* **21** (3): 62–70, May/June 2004.

[25] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, P. America, Generalizing a Model of Software Architecture Design from Five Industrial Approaches, Proc. WICSA 2005, pp. 77-88, IEEE Computer Society 2006.