A State-of-the-Practice Survey of Risk Management in Development with Off-the-Shelf Software Components

Jingyue Li, *Member*, *IEEE Computer Society*, Reidar Conradi, *Member*, *IEEE*, Odd Petter N. Slyngstad, *Student Member*, *IEEE*, Marco Torchiano, *Member*, *IEEE Computer Society*, Maurizio Morisio, *Member*, *IEEE Computer Society*, and Christian Bunse

Abstract—An international survey on risk management in software development with Off-the-Shelf (OTS) components is reported upon and discussed. The survey investigated actual risk-management activities and their correlations with the occurrences of typical risks in OTS component-based development. Data from 133 software projects in Norway, Italy, and Germany were collected using a stratified random sample of IT companies. The results show that OTS components normally do not contribute negatively to the quality of the software system as a whole, as is commonly expected. However, issues such as the underestimation of integration effort and inefficient debugging remain problematic and require further investigation. The results also illustrate several promising effective risk-reduction activities, e.g., putting more effort into learning relevant OTS components, integrating unfamiliar components first, thoroughly evaluating the quality of candidate OTS components, and regularly monitoring the support capability of OTS providers. Five hypotheses are proposed regarding these risk-reduction activities. The results also indicate that several other factors, such as project, cultural, and human-social factors, have to be investigated to thoroughly deal with the possible risks of OTS-based projects.

Index Terms—Software engineering/reusable software, software engineering/management, software engineering/software engineering process.

1 Introduction

Long-term success in commercial software development is becoming increasingly challenging. Clients of development companies expect software systems that follow industrial standards, are of high quality, are reasonably priced, and have a short time to market. Hence, software companies are finding it increasingly difficult to protect their investments in technology and to maintain productivity levels. As a result, the software industry is turning to approaches that promise a larger than normal return on investment. Component-based software development (CBSD) [1] is believed to be one such approach because it is based on the idea of developing/buying once and (re)using often. By reusing Commercial-Off-the-Shelf (COTS) and Open Source Software (OSS) components (collectively called *OTS components*), in addition to components made in-house, the expected benefits increase even more. Unfortunately, using external components introduces a number of additional risks. One is the fact that the functionality and quality of a candidate component may be unknown. Another is the market-related instability of a component provider, who may terminate maintenance support [2], [3], [4], [5]. Thus, project managers have to identify and evaluate possible risks before deciding to acquire an external component instead of developing and reusing an in-house component. Several risks and risk-reduction activities related to OTS-based development have been identified by previous case studies [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12] (a risk-reduction activity is an activity that is designed to minimize a particular risk or group of risks, i.e., to minimize the probability that a problem corresponding to the risk(s) will occur). However, few subsequent large-scale empirical studies have investigated whether these risks do really exist or whether the risk-mitigation strategies really work. As a result, software project managers have only a few effective and well-proven guidelines for assessing the various risks and for deciding upon effective methods of reducing them. Our study is the first to examine empirically the effectiveness of proposed risk-reduction activities of OTS-based development in the industry. We consider a scenario concerning the (re)use of OTS components such that they will be integrated into a new software system or product.

The background of the survey presented in this paper was inspired by a qualitative study [13] in 2002 of COTS usage in seven IT companies in Norway and Italy. The study identified six "theses" on COTS usage, which partly challenged commonly held beliefs. To build upon this

- *J. Li, R. Conradi, and O.P.N. Slyngstad are with the Software Engineering Group, Department of Computer and Information Science (IDI), Norwegian University of Science and Technology, Sem Saelandsvei 7-9, NO-7491, Trondheim, Norway. E-mail: {jingyue, conradi, oslyngst}@idi.ntnu.no.*
- *M. Torchiano and M. Morisio are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi, 24, I-10129 Torino, Italy. E-mail: {marco.torchiano, maurizio.morisio}@polito.it.*
- *C. Bunse is with the International University in Germany, Campus 2/ School of IT, 76646 Bruchsal, Germany. E-mail: Christian.Bunse@i-u.de.*

Manuscript received 10 July 2006; revised 29 Dec. 2007; accepted 8 Jan. 2008; published online 25 Feb. 2008.
Recommended for acceptance by A. Mockus.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0162-0706.
Digital Object Identifier no. 10.1109/TSE.2008.14.

previous study, we first conducted a qualitative prestudy [14] in 2003 by performing structured interviews. We interviewed 16 industrial developers from 13 IT companies to summarize lessons learned from COTS-based development. The principal insights of this prestudy were 1) that COTS-related activities can be integrated successfully into most development processes (incremental and waterfall) and 2) that components are habitually selected in an ad hoc manner, using either in-house expertise or Web-based search engines. This led us to conclude that there is a need to manage component-related experience in a more systematic manner.

The study presented herein is a quantitative follow-up of the qualitative prestudy [14]. It was performed from May 2004 to August 2005 and addressed IT companies in Norway, Italy, and Germany. The survey performed retrospective investigations regarding the process improvement and risk management in 133 completed OTS-component-based projects. Parts of the study's results have been published before. In [15], we report and discuss the actual processes of selecting and integrating OTS components. In [16], we present the differences between using COTS and OSS components. In [17], we categorized and solicited postopinions on 10 main findings and briefly investigated possible explanations for these findings by qualitative phone interviews. The current paper will concentrate on the statistical analysis of the problems that actually occurred and their associated risk-reduction activities. The research questions are the following:

- **RQ1**: Which problems occurred more frequently than others?
- **RQ2**: Which risk-reduction activities were performed most frequently?
- **RQ3**: Which risk-reduction activities were deemed effective for avoiding particular risks?

It was, of course, a necessary precondition for addressing RQ1 that we first identify the problems that occurred. The results showed that the most frequent problems in OTS-based development are the underestimation of effort and the costly identification of the location of defects. It is worth noting that problems regarding the quality of OTS components do not occur frequently, which is what is commonly expected. The results also illustrate several promising risk-reduction activities to ensure the success of OTS component-based development, such as the following: study and understand the OTS components completely, evaluate their quality thoroughly, maintain a continual watch on the reputation of providers and their ability to provide support, limit the number of different components used in one project, and manage and share the company's knowledge on OTS components. Several hypotheses have been proposed based on the effectiveness of these risk-reduction activities upon certain risk items.

The remainder of the paper is organized as follows: Section 2 presents related work. Section 3 discusses the research design and related research questions. Profiles of the collected samples are presented in Section 4. Section 5 presents the empirical results of the research questions. Section 6 contains discussions of the results. Section 7 concludes, notes unresolved issues, and states intentions for further work.

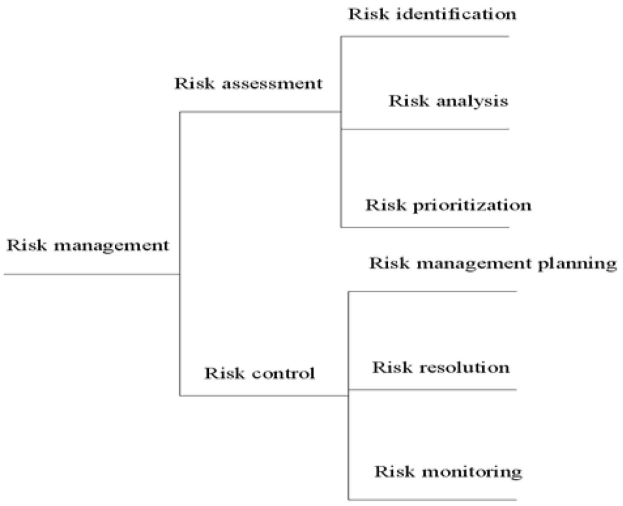

Fig. 1. Risk management framework.

2 Related Work

2.1 Risk Management in Software Development

Risks are factors that may affect a project adversely unless project managers take appropriate countermeasures. Risk management includes two primary activities, risk assessment and risk control, each of which has three subsidiary steps, as described by Boehm [15] and shown in Fig. 1.

Risk identification produces lists of project-specific risk items that are likely to compromise the success of a project. Several lists of possible risks in software projects have been compiled [19], [20], [21] and classified [22], [23], [24], [25], [26]. *Risk analysis* and *risk prioritization* rank the identified risk items by assessing the probability and severity of the loss for each risk item. *Risk management planning* defines how risk reduction will be conducted in a particular project by defining, among other things, risk-reduction tasks, responsibilities, activities, and budget. Risk-reduction activities must take into account the process [27], organization [24], [28], and technology [29]. *Risk resolution* produces a situation in which the risk items are eliminated or otherwise resolved. *Risk monitoring* involves tracking the progress of a project toward resolving its risk items and performing corrective activities where appropriate.

Risk management is particularly important in software development projects due to the inherent uncertainties that most software projects face. Project management has to anticipate risks, understand their impact on the project, and take steps to avoid them [30, p. 105]. Although previous studies have identified several risk items [19], [20], [21], [22], [23], [24], [25], [26], different risk taxonomies may be required by different project contexts because building one complete and universally applicable risk taxonomy is quite unrealistic [31]. Although several risk-reduction strategies have been proposed [24], [27], [28], [29], there is a growing demand for empirical research to provide information about the effectiveness of proposed risk-reduction tools and techniques [32].

TABLE 1
Studies on Risk and Risk Management in OTS Component-Based Development

Study unit	Result/finding	Examples*
COTS in general[1]	Identified the specific challenges of using COTS software by comparing with everything built in-house [35].	Choosing the wrong COTS software may be more expensive than building it in-house.
	Summarized lessons learned from COTS-based university projects and proposed corresponding risk-management strategies [3].	Overly optimistic COTS learning curve (a more likely learning curve must be assessed during planning and scheduling).
	Summarized 10 risk factors in COTS-based development and proposed risk-reduction activities [8].	Risk factor: rapid and asynchronous changes. Risk factor: different quality practices.
	Summarized COTS lessons learned from several industrial case studies [36].	Choosing an appropriate vendor is important for success.
OSS in general[2]	Identified possible problems of development with OSS in industrial settings [9] [12] [37].	If the OSS-based system has no proper license permission, the licensor may claim damages.
CBSD [3]	Identified challenges to, and inhibitors of, component-based development [38].	Component-based development lacks engineering methods.
COTS component[4]	Identified and classified possible risks in COTS component-based development in different development phases, and proposed corresponding risk-management strategies to mitigate the risks [2].	COTS components may have asynchronous update cycles (try to involve a senior analyst in the analysis of the updates and derive an effective field upgrade procedure and schedules).
	Summarized challenges in COTS component-based development [39].	There is insufficient information to evaluate COTS component in the market.
	Summarized problems and good practices in COTS component-based development [14].	Problem that occurred: the learning effort was underestimated. Good practice: do integration testing early and incrementally.

*Examples of the proposed risk-management strategies are in brackets.
[1]COTS in general: any software bought from third-party companies.
[2]OSS in general: any software acquired from the Open Source community.
[3]CBSD: using components without differentiating between in-house built, COTS, or OSS components.
[4]COTS component: software component bought from third-party companies

2.2 Specific Risk Management in Development with OTS Components

When doing research on component-based development, the most prevalent issue concerns what constitutes a component. Of course, there are standard definitions, such as those provided in [33] or [34]. However, these definitions might differ from the view of component marketplaces, such as ComponentSource.com and SourceForge.net. In order to define a common basis within this study, we used the following definitions:

- **Component.** This is a program unit of independent production, acquisition, and deployment that can form part of a functioning system. We limit ourselves to components such that it has been decided explicitly to build them from scratch or to acquire them externally as OTS components. That is, such components are not shipped with the operating system, not provided by the development environment, and not included in any preexisting platform. This means that platform ("commodity") software, e.g., Linux, DBMSs, various servers, and similar software, is not considered. Furthermore, components usually follow some component model, as expressed by the COM, CORBA, or EJB; alternatively, they can be a C++ or Java library.
- **OTS component.** This is a component that is either provided by a commercial COTS vendor or obtained from the open source community. OTS components may come with certain obligations, e.g., payment or licensing terms. Furthermore, control over their features and evolution lies mostly with the component provider. The source code of an OTS component is not usually modified, even if it is available.

OTS component-based development encompasses three primary types of stakeholder: component developers, application assemblers/integrators, and clients. Different stakeholders might face different risks and experience different challenges [11]. This study focuses on risk management for OTS-based development. Hence, we only investigate those risks that are relevant to the assemblers or integrators of the final software application. Further, given that the term OTS component covers both COTS and OSS components, we limit ourselves to risks and risk-reduction activities that address common issues in component-based development, COTS-based development, and OSS-based development.

Several studies have been conducted to investigate the challenges, risks, and risk-management strategies involved in developing software with OTS components (see Table 1 for a summary of their findings).

An examination of their results reveals that the risk items that were identified are based on either common sense assumptions [35] or small, nonrepresentative case studies [8], [9], [12], [36], [37]. Without evidence from large-scale empirical studies, it is difficult to convince practitioners that the risk items that were identified constitute typical problems that need to be addressed prior to starting a project. Without knowing the frequency with which typical problems occur, it is difficult for practitioners to estimate the likelihood that identified risk items will result in corresponding problems and to prioritize the risks. Furthermore,

TABLE 2
Typical Risks in OTS Component-Based Development

Stage	Issue	Risk item	Description
Selection and integration	Project manag.	Selection effort ill-estimated	Effort to select OTS components is not estimated satisfactorily [3].
	Project manag.	Integration effort ill-estimated	Effort to integrate OTS components is not estimated satisfactorily [2].
	Integration	Negative impact on reliability	OTS components affect system reliability negatively [10] [11] 12].
	Integration	Negative impact on security	OTS components affect system security negatively [10] [11] [12].
	Integration	Negative impact on perform.	OTS components affect system performance negatively [10] [11] [12].
	Integration	Not adaptable to req. changes	OTS components cannot be adapted sufficiently to changing requirements [3].
	Integration	Requirements not negotiable	It is not possible to (re)negotiate requirements with the client, if OTS components do not satisfy all requirements [14].
	Integration	Defect location difficult	It is difficult to identify whether defects lie inside or outside the OTS components [3].
	Integration	Deploy unfeasible	OTS components are not sufficiently compatible with the production environment when the system is deployed [11].
Maintenance	Project manag.	Maintenance planning unfeasible	It is difficult to plan system maintenance, because different OTS components have asynchronous release cycles [2].
	Project manag.	Upgrade unfeasible	It is difficult to upgrade the system with the last OTS component version [2].
	Vendor relation	Lack of information on provider	Information about the reputation and technical support ability of the provider is inadequate [2] [9].
	Vendor relation	Lack of support	The provider does not provide sufficient technical support/ training [2] [9].

most of the risk-reduction strategies that have been proposed are derived from lessons learned or best practice know-how. Best practices may contribute positively to the project as a whole [14]. However, it is not clear which particular best practice might serve to reduce the probability that a problem corresponding to a particular risk item will occur during the project. Thus, further investigation is required if the preconditions and possible benefits of using best practices are to be identified. In addition, different risk-reduction strategies may have been proposed to reduce the likelihood that a risk item will occur as a problem. For example, to reduce the risk of selecting an improper OTS component, components may be selected on the basis of project requirements and negotiation with clients [2], [3]. Another proposed strategy for reducing this risk is to focus more on matches with the proposed architecture than on requirements [13]. Without evidence from empirical studies that compare the effectiveness of different strategies, it is difficult for practitioners to focus on the most cost-effective risk-reduction activities.

3 Research Design

In practice, the creation of a plan for managing a project requires project managers to first estimate the likelihood that a particular risk will arise during the project, how much effort is needed to perform a specific risk-reduction activity, and how risks might be systematically managed. In order to help project managers to predict the probability that a particular risk will arise during the project and to plan risk management efficiently, it is necessary to summarize the practices and results of risk-reduction activities from past projects (e.g., in the form of guidelines that may be tailored to different organizations). This being so, the motivation for the research is to perform a posteriori measurements and to collect experience of both actual project risks and the effect of the risk-reduction activities that were performed. This study is designed to be exploratory, to report on the state of the practice, and to generate hypotheses for further testing.

3.1 Research Questions

As mentioned in Section 2, previous studies identified risk items on the basis of a few case studies, without knowing whether the identified risks occurred frequently or happened incidentally. Our first research question, **RQ1**, was designed to determine the relative frequency with which particular risks were present in various OTS component-based development projects. In general, the abstract OTS-based development process consists of the following phases [40]: assessment and selection of OTS components, tailoring and integration of OTS components, and maintenance of OTS and non-OTS parts of the system. Each phase of OTS-based development has risks, which can be classified as relating to integrating the application (e.g., satisfying requirements), project management (e.g., quality control), and buying (e.g., vendor relationship) [11]. To identify risks that pertain to software development with OTS components, we performed a review of the literature, which resulted in selecting a number of publications [2], [3], [10], [11], [12], [14]. These papers listed risk items on the basis of experience or lessons learned. We believe that these items may occur frequently in practice and warrant further investigation. From the publications, we selected those risk items (see Table 2) that satisfy the following criteria:

- The risk item must be a common issue in all of CBSD, COTS, and OSS-based development. This resulted in the exclusion of a number of items. For example, risks relevant to OSS licensing [12] were excluded because they rarely occur in COTS-based development.
- The risk item should be relevant only to the assemblers or integrators of the application. This also resulted in a number of items being excluded, such as risks related to the providers of components in [11].

As discussed in Section 2, the majority of published risk-reduction strategies are based on a few case studies with specific contexts. It is therefore important to demonstrate that the strategies were performed properly. The purpose of our second research question, **RQ2**, was to investigate the

TABLE 3
Typical Risk-Reduction Activities in OTS Component-Based Development

Stage	Risk management activity	Description
Selection and integration	Client attends decision	Client was actively involved in the "acquire" vs. "build" decision of OTS components [8] [14].
	Client attends selection	Client was actively involved in the selection of OTS components [2] [3] [8].
	Architecture-based selection	OTS components were selected mainly on the basis of architecture and compliance with standards, instead of expected functionality [13].
	Thorough quality evaluation	The characteristics of OTS components with respect to quality (reliability, security etc.) were considered seriously in the selection process [2] [3] [14].
	Add learning effort	Effort required to learn OTS components was considered seriously in effort estimation [3].
	Add testing effort	Effort expended in black-box testing of OTS components was considered seriously in effort estimation [3] [14].
	First integrate unfamiliar component	Unfamiliar OTS components were integrated first [14].
	Incremental testing	Integration testing was done incrementally (after each OTS component was integrated) [14].
Maintenance	Follow component update	Local OTS experts actively followed updates to, and possible consequences of integrating, OTS components [14].
	Trace substitute component	Maintained a continual watch on the market and looked for possible substitute components [14].
	Trace provider	Maintained a continual watch on the reputation of providers and their ability to provide support [2].

application of risk-reduction activities in actual industrial projects. We selected a set of risk-reduction strategies (see Table 3) from the the literature [2], [3], [8], [13], [14] using the same processes and criteria as for the risk items.

The purpose of the third research question **RQ3** was to examine the correlation between the level of risk presence and the degree to which risk-reduction activities were adopted. If we found a significant negative correlation between the performed risk-reduction activity and the actual occurrence of a certain risk, we assumed that the associated risk-reduction activity had been effective. For a specific risk, we were interested in the ***most effective risk-reduction activity***.

In addition, we hypothesized that the capability of coping with the project risks is also contingent upon several factors that pertain to the context of a project. These include business characteristics (time-to-market pressure and project budget), technology characteristics (project complexity and quality requirements), and organization characteristics (project members' experience of OTS-based development). Thus, we sought to understand the impact of specific risk mitigation strategies by controlling these environmental factors.

3.2 Survey Design

We used a structured questionnaire to collect data. The questionnaire had five major parts. The first part was introductory and defined our unit of study as a completed software development project that uses one or more OTS components. The participants were asked to select one such project that they were most familiar with as a basis for completing the questionnaire. The second part provided definitions of key concepts used in the questionnaire, such as the definition of an OTS component. The third part was aimed at collecting background information about the company, actual ("chosen") project, and respondents. The fourth part contained detailed questions, especially on risk management and process improvement. The last part was aimed at collecting more detailed information about one particular component that was used in the project. Data related to research questions herein were collected by asking participants to respond to the following questions, in most cases using a five-point Likert scale [41]:

- The first question concerned the risk items in Table 2. Participants were asked "Did risk factor x occur?" The alternatives provided to the participants were "do not agree at all," "hardly agree," "agree somewhat," "agree mostly," "strongly agree," or "do not know," with 1 being assigned to "do not agree at all" and 5 being assigned to "strongly agree."
- To collect data concerning the risk-reduction activity variables, we listed possible risk-reduction activities, as shown in Table 3, and asked participants whether such activities had been performed, providing the same alternatives as those used for the risk items.
- To measure the respondents' views regarding the business characteristics of the project, we asked participants about the time to market and cost/effort emphasis of the project. The alternatives provided were "very low," "low," "medium," "high," "very high," or "do not know," with 1 being assigned to "very low" and 5 being assigned to "very high."
- To measure the respondents' views regarding the technology characteristics of the selected project, we used the same alternatives as for business characteristics and asked the participants to rate the reliability, security, and performance requirements of the final system/product. Given that the number of different COTS packages used in one project may influence the complexity of the project dramatically [42], we also asked the respondents to state the number of different OTS components that were used.
- To measure the organizational characteristics of the selected project, we asked participants to state what percentage of project members had previous experience with OTS-based development.

3.3 Data Collection

We used stratified random sampling to select companies and convenience sampling to select projects inside a company. The survey was performed in Norway, Italy, and Germany. This study is probably the first software engineering survey

to use random census-type data, such as data from the Norwegian Census Bureau (SSB) [43]. We found that the entire sampling and contact process can be unexpectedly expensive. Given a person-year of total effort just in Norway, the cost of a single completed questionnaire is about one person-week. The study reveals that, at best, we can achieve a stratified-random sample of IT companies, followed by a convenience sample of relevant projects. All respondents in Norway and Italy used the Web-based Simula Experiment Support Environment (SESE) [44] tool to complete the questionnaire. A number of questionnaires were completed by the researcher in Germany through telephone interviews because of the concerns of some companies regarding confidentiality. Details of the sample selection, contact process, and response rate are reported in the Appendix and discussed in [45].

3.4 Data Analysis

Although the variables in this study were measured using a five-point Likert scale, we treated them as interval variables. Spector [46] has shown that people tend to treat categories in Likert scales as equidistant, regardless of the specific categories employed. In addition, using parametric tests for scales that do not contain strictly equal intervals does not lead, except in extreme cases, to incorrect statistical conclusions [47]. Hand concluded that restrictions on statistical operations that arise from the type of scale used are more important in contexts that pertain to model fitting and hypothesis testing than in those that pertain to the generation of models or hypotheses [48]. In the latter contexts, in principle, anything is legitimate in the initial search for potentially interesting relationships [48].

For research questions **RQ1** and **RQ2**, we analyzed the level of presence of risks and level of adoption of risk-reduction activity using the *median* and *mode*. In addition, we analyzed the intercorrelations between occurrences of risk items and risk-reduction activities, respectively.

For research question **RQ3**, we viewed the level of the occurrence of risk items as a dependent ("success") variable and the various risk-reduction activities as independent variables. Hence, for each risk item shown in Table 2, we were interested in the relative predictive importance of the risk-reduction activities displayed in Table 3. The risk items in Table 2 and risk-reduction activities in Table 3 are categorized as belonging to either the selection and integration phase or the maintenance phase. For the risk items and risk-reduction activities of each phase, we first used Pearson's correlation analysis (excluding missing data pairwise) [49, p. 242] to identify those risk-reduction activities that have significant ($p < 0.05$) negative correlations with a certain risk item. If a risk item was correlated significantly with more than one risk-reduction activity, we used stepwise multiple regressions [49, p. 532] to compare the effectiveness of those risk-reduction activities. In the first stage of the stepwise multiple regression, the risk-reduction activity that is best correlated with a certain risk item was included in the regression equation. In the second stage, the remaining risk-reduction activities with the highest partial correlation with the risk item were entered. This process was repeated, at each stage addressing previously entered risk-reduction activities, until the addition of a remaining risk-reduction activity did not increase R^2 by a significant amount (or until all variables are entered).

For research question **RQ3**, further analysis is needed to identify the possible effects of contextual factors on the correlations between risk items and the most effective risk-reduction activities. Since we were concerned that the correlation between the risk items and the risk-reduction activities may be partially confounded by different project contexts, we asked the question "*What is the correlation between risk items and the risk-reduction activities when the project context variables are controlled?*" We used hierarchical multiple regressions [49, p. 523] to examine the effects of each project context variable. For each risk item, a project context variable was entered in the first block of the hierarchical multiple regression analysis. The corresponding most efficient risk-reduction activity (i.e., the result of stepwise multiple regressions) was entered in the second block of the hierarchical multiple regression analysis. The results for the second block told us how much additional variation in the risk item is accounted for by the corresponding risk-reduction activity once the effect of the project context has been controlled.

4 Collected Data

We gathered results from 133 projects (47 from Norway, 48 from Germany, and 38 from Italy) in 127 companies. In general, we selected one project from each company. However, we selected more than one project from three Norwegian IT companies because those companies had many OTS-based projects and wanted to contribute more to this study.

The respondents came from small (29 percent), medium (31 percent), and large (40 percent) companies. The main business areas of the sampled companies were as follows: software house (50 percent), IT consulting company (32 percent), telecommunication industry (2 percent), and IT department of other industries (16 percent). The final systems produced by these 133 projects are deployed in various business domains: public sector (25 percent), banking or finance (16 percent), manufacturing industry (31 percent), IT sector (9 percent), and others (19 percent).

In the 133 projects that were selected, 83 used only COTS components, 44 used only OSS components, and six used both. The mean value of the number of different OTS components used in a project was three (median value was two). The mean effort used before the first delivery of a project was 86 person-months and the median value was 32 person-months.

Most respondents of the 133 projects had a solid IT background. More than 90 percent of them were IT managers, project managers, or software architects. Most of them had more than two years of experience with OTS-based development. All of them held at least a bachelor's degree in computer science or telecommunication.

The OTS components that were used in the investigated projects included components that follow the COM, DCOM, and EJB models. There were also some components in the format of C++ or Java libraries. Examples of the selected components are listed in Table 4.

TABLE 4
Examples of OTS Components

Name of OTS component	OSS/COTS	Functionality	Component model/technique
Log4j	OSS	Logging and tracing	Library
Snoopy	OSS	RSS parser	PHP Library
Crystal Report	COTS	Report design and development	J2EE, .NET, COM
MapObjects	COTS	Collection of embeddable mapping and GIS components	Library

TABLE 5
Risk Occurrences

Risk item	Number of answers (valid/missing)	Median	Mode
Selection effort ill-estimated	128/5	2	2
Integration effort ill-estimated	130/3	3	3
Negative impact on reliability	117/16	1	1
Negative impact on security	98/35	1	1
Negative impact on perform.	121/12	1	1
Not adaptable to req. changes	131/2	2	2
Requirements not negotiable	123/10	2	1
Defect location difficult	129/4	3	3
Deploy unfeasible	123/10	2	2
Maintenance planning unfeasible	123/10	2	1
Upgrade unfeasible	129/4	2	1
Lack of information on provider	119/14	2	1
Lack of support	112/21	2	1 and 2

5 Results for the Research Questions

5.1 RQ1: Occurrences of Risks in OTS Component-Based Projects

The median and mode value of the risk occurrences (excluding missing data) are presented in Table 5 and the correlation (excluding missing data pairwise) between them are shown in Table 6. The data in Table 5 show that 11 out of 13 risks occurred infrequently. Among them, the least frequent risks are the possible negative impacts of OTS components on the quality of the whole system, such as reliability, security, and performance. Surprisingly, 26 percent of the answers to the risks related to security were missing. This probably indicates that few respondents are able to assess or care about the security effects of the OTS components on the whole system. Two risks, i.e., imprecise estimation on the integration effort and costly defect localization, occurred more frequently than others.

By excluding insignificant correlations ($p > 0.05$) and correlations with coefficient smaller than 0.30, the remaining data (marked by gray) in Table 6 show that imprecise effort estimation on selection and on integration are correlated strongly. In addition, occurrences of quality-related risks are intercorrelated strongly. The same applies to maintenance-related risks. Moreover, the difficulties of following requirement changes are strongly correlated with the occurrence of imprecise selection effort estimation and poor system quality. Costly debugging is strongly correlated with the unsatisfactory estimation of the effort required for integration and costly system deployments.

5.2 RQ2: Performed Risk-Reduction Activities

The median and mode of the performed risk-reduction activities (excluding missing data) are shown in Table 7. The correlations (excluding missing data pairwise) between occurrences of risk-reduction activities are presented in Table 8.

The data in Table 7 illustrate that seven out of 11 risk-reduction activities were performed infrequently. Activities relevant to the clients, such as involving such a stakeholder in the "acquire versus build" decision procedure or in the actual selection of OTS components, were rarely performed. Some risk-reduction activities were performed more frequently. For example, developers usually seriously evaluated the quality of the OTS components during the selection phase. Some developers tried to perform integration testing as early as possible and integrated the unfamiliar components first. To maintain the system successfully, some developers also consulted internal experts to follow the updates of an OTS component.

By excluding insignificant correlations ($p > 0.05$) and correlations with a coefficient smaller than 0.30, the remaining data (marked by gray) in Table 8 show that the clients' involvement in selecting OTS components is correlated strongly with their participation in the decision to use OTS components. In addition, the granted learning and testing effort is strongly correlated with the emphases on evaluating the quality of components. Furthermore, developers' activities on tracing component providers' support reputation and capability are strongly correlated with their activities on tracing information about the chosen component.

5.3 RQ3: Risks and Risk-Reduction Activities

The significant negative correlations between risk items and the risk-reduction activities revealed by Pearson's correlation analysis are summarized in Table 9. The effectiveness of risk-reduction activities was compared with *stepwise multiple regression analysis* (significant at the 0.05 level, including constants in of the equations and excluding missing data listwise). All of our regression models satisfy

TABLE 6
Correlations between Risk Occurrences

Risk Items	Integration effort ill-estimated	Negative impact on security	Negative impact on perform.	Not adaptable to req. changes	Req. not negotiable	Defect location difficult	Deploy unfeasible	Upgrade unfeasible	Lack of information. on provider	Lack of support
Selection effort ill-estimated	.614 (.000) [128]			.370 (.000) [127]			.196 (.033) [119]			
Integration effort ill-estimated				.286 (.001) [129]		.336 (.000) [126]				
Negative impact on reliability		.581 (.000) [97]	.559 (.000) [114]	.430 (.000) [115]	.297 (.002) [107]		.295 (.002) [108]			
Negative impact on security			.424 (.000) [96]	.342 (.001) [97]	.226 (.033) [89]					
Negative impact on perform.				.239 (.009) [119]						
Not adaptable to req. changes					.209 (.021) [122]	.201 (.023) [128]	.292 (.001) [122]			
Defect location difficult							.446 (.000) [120]			
Maintenance planning unfeasible								.312 (.000) 120	.227 (.010) [112]	.227 (.020) [105]
Upgrade unfeasible									.371 (.000) [116]	.180 (.061) [109]
Lack of information on provider										.306 (.002) [105]

Numbers in () are p-values.
Numbers in [] are numbers of cases.

TABLE 7
Occurrences of Risk-Reduction Activities

Risk management activity	Number of answers (valid/missing)	Median	Mode
Client attends decision	131/2	1	1
Client attends selection	129/4	1	1
Architecture-based selection	131/2	2	2
Thorough quality evaluation	130/3	3	5
Add learning effort	129/4	3	2
Add testing effort	124/9	2	2
First integrate unfamiliar component	115/18	3	3
Incremental testing	127/6	3	3
Follow component update	131/2	3	3
Trace substitute component	130/3	2	1 and 2
Trace provider	125/8	2	1

the basic assumptions [49, p. 533] of multiple regressions. The results of the stepwise multiple-regression analyses are presented in Table 10 and show the following:

- Thorough quality evaluation of a component in the selection phase is correlated negatively with occurrences of most development-related risks.
- Devoting a significant portion of effort to learning about an OTS component is correlated negatively with the occurrences of several risk items, such as imprecise estimation of the selection and later integration effort and poor efficiency in following requirement changes and in locating defects.
- Integrating unfamiliar components first is correlated negatively with inefficient requirement negotiation. However, it is correlated positively with inefficient follow-up of requirement changes.
- Monitoring the support reputations of a provider is negatively correlated with the risk of lacking support from the provider.
- However, certain risk items, such as negative impact on performance, negative impact on security, and difficulty in planning later maintenance, are not correlated negatively with any risk-reduction activities.

TABLE 8
Occurrences of Risk-Reduction Activities

	Client attends selection	Thorough quality evaluation	Add learning effort	Add testing effort	Incremental testing	Follow component update	Trace substitute component	Trace provider
Client attends decision	.799 (.000) [127]	.179 (.043) [128]	.225 (.011) [127]	.219 (.015) [122]				
Client attends selection		.221 (.013) [126]	.182 (.042) [125]					
Thorough quality evaluation			.428 (.000) [128]	.259 (.004) [122]	.180 (.045) [125]			
Add learning effort				.545 (.000) [122]	.180 (.045) [124]			
Add testing effort					.212 (.020) [119]			
Follow component update							.224 (.011) [129]	.368 (.000) [124]
Trace substitute component								.545 (.000) [124]

Numbers in () are p-values.
Numbers in [] are numbers of cases.

The effect of the project context variable was tested using hierarchical multiple regressions, as explained in Section 3.4. The project context variables explain significant variances of a certain risk item (i.e., project context variables with a significant R^2 change) and the additional variation accounted for by the corresponding most efficient risk-reduction activity (i.e., the result of stepwise multiple regressions shown in Table 10) stand in Table 11, which illustrates the following:

- The correlations between risk items and the risk-reduction activities in Table 10 are still valid when the project context variables are controlled.
- The general experience of OTS component-based development by project members is correlated negatively with imprecise effort estimation, inefficient debugging, and costly system deployment. However, the number of different components used in the project is correlated positively with costly system deployment, maintenance, and debugging.
- It is not surprising that strict reliability requirements are correlated positively with the reliability of the system. One possible explanation is that project members may put more effort into quality control when the project is reliability critical.
- Surprisingly, the experience of the developers is correlated negatively with their satisfaction with the technical support from the provider. One possible explanation is that experienced developers may have higher expectations of provider support than less experienced people.
- The maximum R^2 of each stepwise regression was, in total, 0.147. This shows that our investigated risk-reduction activities explain less than 15 percent of the variance of each corresponding risk. This probably means that more efficient risk-reduction activities had not been noted in the literature and, hence, were not included in our study.

6 Discussion

6.1 Comparison with Related Studies

It is difficult to estimate the effort expended on selecting OTS components and on integrating them [3], [50]. Basili and Boehm [51] hypothesized that COTS-based development is currently a high-risk activity, with effort and schedule overruns exceeding non-COTS-based software overruns. The results of **RQ1** show that inaccurate estimation of integration effort occurs more frequently than other problems. Kotonya and Rashid [7] point out that several OTS components with similar functionalities may have very different system resource requirements. When a problem is identified, it may not be obvious where its root cause lies: It may lie in one of the externally acquired components, in the integration code (gluecode), or it could be due to a misinterpretation of a component interface [2]. The results of **RQ1** show that one significant problem regarding OTS components is the high cost of locating defects.

One proposed strategy for reducing risks when selecting suitable OTS components is to select components on the basis of requirements and (re)negotiation with clients [2], [3]. However, the results for **RQ2** show that very few projects involved their clients in the selection of OTS components. Another proposed strategy for reducing risks is to focus more on architectural matches than the fulfillment of requirements [13]. However, the results for **RQ3** do not reveal any correlation of this strategy with the risk items investigated in our study.

TABLE 9
Correlations between the Risk Items and Risk-Reduction Activities

Risk items	Risk-reduction activities in the selection and integration phases								Risk-reduction activities in the maintenance phase		
	Client attends decision	Client attends select.	Architectu re-based selection	Thorough quality evaluation	Add learnin g effort	Add testing effort	First integrate unfamiliar comp.	Increm ental testing	Follow comp. update	Trace substitute comp.	Trace provider
Selection effort ill-estimated		-.177* (0.49) [124]		-.203* (.023) [126]	-.195* (.029) [126]						
Integration effort ill-estimated				-.231* (.009) [128]	-.211* (.017) [127]						
Negative impact on reliability				-.273** (.003) [114]				-.224* (.018) [111]			
Negative impact on security											
Negative impact on perform.											
Not adaptable to req. changes				-.252** (.004) [129]	-.198* (.025) [128]		**.198* (.034) [115]**				
Req. not negotiable							-.249** (.009) [109]				
Defect location difficult				-.189* (.033) [127]	-.211* (.018) [126]						
Deploy unfeasible				-.283** (.002) [120]							
Maintenance planning unfeasible											
Upgrade unfeasible											
Lack of information on provider											
Lack of support											-.232* (.016) [107]

** Correlation is significant at the 0.01 level (2-tailed). *Correlation is significant at the 0.05 level (2-tailed).
Numbers in () are p-values. Numbers in [] are numbers of cases.

TABLE 10
The Risks and Corresponding Most Effective Risk-Reduction Activities

Risk items	Risk-reduction activities		
	Most effective risk-reduction activity	**Total adjusted R^2 (df1/df2)**	**Standardized Beta**
Selection effort ill-estimated	Thorough quality evaluation	.039* (1/119)	-.217
Integration effort ill-estimated	Thorough quality evaluation	.047* (1/124)	-.234
Negative impact on reliability	Thorough quality evaluation	.068** (1/107)	-.276
Negative impact on security			
Negative impact on perform.			
Not adaptable to req. changes	Thorough quality evaluation	.056** (1/127)	-.252
Not adaptable to req. changes	First integrate unfamiliar component	.050* (1/112)	.241
Req. not negotiable	First integrate unfamiliar component	.053** (1/107)	-.249
Defect location difficult	Add learning effort	.040* (1/123)	-.218
Deploy unfeasible	Thorough quality evaluation	.087** (1/118)	-.308
Maintenance planning unfeasible			
Upgrade unfeasible			
Lack of information on provider			
Lack of support	Trace provider	.045* (1/105)	-.232

** Significance of F change is less than 0.01. * Significance of F change is less than 0.05

TABLE 11
The Effect of the Project Context Variables

Risk items	Project context variable			Risk-reduction activities			Total adjusted R^2 (df1/df2)
	Name	Standard Beta	Adjusted R^2 change	Name	Standard Beta	Added R^2 change	
Selection effort ill-estimated	Experience	-.294	.080**	Thorough quality evaluation	-.197	.032*	.112 (1/122)
Integration effort ill-estimated	Experience	-.328	.103**	Thorough quality evaluation	-.224	.044*	.147 (1/124)
Negative impact on reliability	Reliability	-.162	.046*	Thorough quality evaluation	-.217	.034*	.080 (1/110)
Not adaptable to req. changes	Reliability	-.120	.036*	Thorough quality evaluation	-.209	.029*	.065 (1/124)
Not adaptable to req. changes	Reliability	-2.36	.032*	First integrate unfamiliar component	.229	.044*	.076 (1/111)
Req. not negotiable				First integrate unfamiliar component	-.249	.053**	.053 (1/107)
Defect location difficult	Experience	-.258	.039*	Add learning effort	-.262	.060*	.099 (1/122)
	Number	.164	.024*	Add learning effort	-.186	.027*	.051 (1/119)
Deploy unfeasible	Experience	-.182	.033*	Thorough quality evaluation	-.272	.066*	.099 (1/116)
	Number	.192	.034*	Thorough quality evaluation	-.276	.069*	.103 (1/114)
Lack of support	Experience	.226	.067*	Trace provider	-.164	.016*	.083 (1/103)
	Number	.242	.056*	Trace provider	-.197	.030*	.086 (1/101)

** Significance of F change is less than 0.01
* Significance of F change is less than 0.05

6.2 Hypotheses for Effectiveness of Risk Reduction Activities

From the results in Tables 10 and 11, we have observed the potential effectiveness of certain risk-reduction activities on corresponding risk items and formulated hypotheses on the basis of these observations.

6.2.1 The Potential Effectiveness of Considering Possible Efforts for Learning OTS Components

Boehm et al. [3] found that one common problem in OTS-based development is the overly optimistic OTS learning curve and propose that a realistic OTS learning curve be determined and taken into account during planning and scheduling. However, the mode value of the variable "additional learning effort" in Table 7 is only 2, which shows that most respondents did not consider the possible learning curve seriously. As found by Rose [2], developers usually work with several OTS components, of which they often have only a partial understanding.

The results for **RQ3** illustrate that devoting a sufficient effort to learning OTS component candidates thoroughly tends to reduce the occurrence of imprecise estimation of project effort. Again, allocating more time for developers to learn and understand OTS components appears to reduce the probability that debugging will be inefficient. Thus, we can formulate the following hypothesis:

Hypothesis 1. Allowing sufficient time for learning during the planning phase will reduce the occurrence of imprecise effort estimation and inefficient debugging.

6.2.2 The Potential Effectiveness of Careful Evaluation of the Quality of OTS Components

For a high-quality system, it is important to ensure, for every system component, that it does not adversely affect the quality of the system as a whole. In the COTS component market, the salesmen may make false, inaccurate, or exaggerated claims about the functionality and other aspects of their products [3], [8], [50]. For OSS components, there is never enough information provided for the integrators to obtain a full picture of the functionality and quality of the components. Thus, the developers themselves need to evaluate the quality of possible OTS components in the selection phase, by either reading the source code or doing black-box testing. The results for **RQ3** show that a thorough evaluation of the quality of OTS components in the selection phase appears to reduce the occurrences of several risks such as a component's negative impact on system reliability, imprecise effort estimation, costly system deployment, and inadequate adaptation after requirement changes. Thus, we can formulate the following hypothesis:

Hypothesis 2. A thorough up-front evaluation of the quality of OTS components will reduce the occurrence of poor system quality and improve system flexibility.

6.2.3 The Potential Effectiveness of Integrating Critical and High-Risk Components First

An important issue with respect to the integration of OTS components concerns changes in requirements and mismatches between the component functionality that is desired and that which is provided [2], [3], [8], [50]. It is difficult to modify an external OTS component to reflect the clients' most recent changes in requirements. Hence, it is important for the OTS component integrator to be able to (re)negotiate requirements with the client [2], [3] and to involve clients in the "acquire versus build" decision procedure and in the actual selection of OTS components [8]. However, the results from **RQ2** show that very few integrators have actually involved clients in these decision-making processes. It was found in [14] that it is not easy to negotiate requirements with the client in OTS-component-based development. One reason is that the client typically cares only about the final product and lacks the willingness or capability to assimilate and assess the technical details. The results from **RQ3** show that integrating the unfamiliar (those high-risk or critical) OTS components first tends to reduce the occurrence of inefficient requirement negotiations. One possible explanation is that integrating such

components first can help to identify problems in the early phases of the project and this may give the integrator enough leeway to (re)negotiate the clients' requirements [52]. However, integrating unfamiliar components first could result in the system being less flexible, which would make it more difficult to follow changes to requirements. Developers may have difficulty in developing familiarity with the system if it involves unfamiliar OTS components. Such lack of familiarity may make it difficult for developers to make dramatic changes to the system in response to changes to the requirements by the client. Thus, we formulate the following hypothesis:

Hypothesis 3. Integrating unfamiliar OTS components first will facilitate requirement (re)negotiation but make it more difficult to modify the system in response to changes to requirements.

6.2.4 The Potential Effectiveness of Monitoring OTS Providers

Maintaining an OTS component-based system is difficult and postdeployment costs may exceed the initial development costs [51]. Different client-vendor release cycles may complicate how often the OTS components in a system should be replaced and it will be difficult to assess the impact of the release of updated components on the rest of the system [2], [7], [8], [50]. Hence, it is important to receive sufficient technical support from the provider. Although the developers have access to the source code for all OSS components (and, in our sample, 1/3 of COTS components), few of them have either read the code or modified it to meet their requirements [53]. It is also advisable to avoid modifying OTS components, even when it is possible [8]. Users of COTS components (here, developers) can get support from the provider by signing a support contract. However, COTS providers may discontinue support for their products for different reasons (for example, the deployed version of a COTS component may be too old to be supported) or may even go out of business [2], [51]. With respect to OSS components, some of them are currently well supported by volunteers in OSS projects. However, current quick support does not ensure that the integrators will receive satisfactory support in the future. In contrast to the situation with COTS components, in which the developers can sign a support contract, the integrators of OSS components have (in principle) no control over future technical support from the OSS community. Our study tried to identify common risk-reduction activities for both COTS and OSS components. Hence, we did not investigate contract issues, which are less applicable for OSS integrators. We devoted our efforts to investigating those risk-reducing activities that can be controlled by almost all OTS integrators. Regarding risks that are related to provider support, the results for **RQ3** show that maintaining a continuous watch on the support reputation of the OTS component providers, as proposed in [8], [50], appears to reduce the occurrence of insufficient support from providers. Thus, we formulate the following hypothesis:

Hypothesis 4. Monitoring the support reputation of an OTS provider will help when selecting a provider and will therefore reduce the danger of insufficient support in the future.

6.3 Hypotheses for Possible Effects of Contextual Factors

Due to the complexities of development with OTS components, not all issues can be documented clearly. Personal capabilities and experience are probably important factors that influence OTS-based development [51]. Donald et al. [42] conclude that the most significant factor that influences the life-cycle cost of a COTS-based system is the number of COTS packages that must be synchronized within a release. The results for **RQ3** illustrate that several contextual factors, such as experience with OTS-based development, the reliability requirements of the system, and the number of different components used, have a confounding effect on the effectiveness of the risk-reduction activities mentioned above. The effect of these context variables needs to be adjusted before testing hypotheses **H1** to **H4**. Thus, we formulate the following hypothesis:

Hypothesis 5. The effectiveness of risk-reduction activities will be confounded by contextual factors such as the experience of the developers, the number of different components, and the requirements on the quality of the system.

6.4 Remaining Issues of Managing Risks in OTS Component-Based Development

The most effective risk-reduction activities that we studied (shown in Table 10) explain less than 10 percent of the variations of the actual risk variable. A further question is how much variance of a risk variable can be explained if the integrators consolidate all risk-reduction activities investigated in this study. For each risk item in the selection and integration phases, we entered all risk-reduction activities into one block of a regression analysis. The same goes for risk items in the maintenance phase. The results show that the maximum variance (R^2) of a risk item that is explained by consolidating all these risk-reduction activities is 0.18. That being so, we may have to look for "non-OTS-specific" risk reduction activities such as code reviews or general process improvement to avoid some risks more effectively. In addition, the integration, debugging, and deployment of OTS components may be eased by employing some advanced technologies for managing the assembly and effective integration of components, e.g., for making statements about components that are to be integrated into a system.

6.5 Possible Threats to Validity

6.5.1 Internal Validity

We promised to send the respondents a final report and to hold a seminar (in Norway) to share our experiences. Our reasons for doing so were to get better feedback and honest answers and to offer a reward for participation. One positive effect of our strategy is that participants may have answered more carefully because they knew that they would get feedback. One possible negative effect is that, since the participants knew that they were being observed, they may have altered their behavior. In general, we believe the participants wanted to share their experience and to learn from others. Thus, we think that the participants answered the questionnaire truthfully. However, the

TABLE 12
The Effect of Country Differences

Risk items	Country code	Risk-reduction activities	
	Adjusted R^2 change	Name	Added adjusted R^2 change (df1/df2)
Selection effort ill-estimated		Thorough quality evaluation	.034* (1/124)
Integration effort ill-estimated		Thorough quality evaluation	.100** (1/126)
Negative impact on reliability	.080**	Thorough quality evaluation	.059** (1/111)
Not adaptable to req. changes		Thorough quality evaluation	.056** (1/127)
Not adaptable to req. changes		First integrate unfamiliar comp.	.031**(1/113)
Req. not negotiable		First integrate unfamiliar comp.	.053** (1/107)
Defect location difficult	.454**	Add learning effort	.020* (1/123)
Deploy unfeasible	.051**	Thorough quality evaluation	.043** (1/117)
Lack of support		Trace provider	.036* (1/105)

** Significance of F change is less than 0.01
*Significance of F change is less than 0.05

projects were selected by the respondents at their convenience. The respondents may have selected the most successful project to use as a basis for completing the questionnaire, although we asked them to use the most familiar one. We controlled for this possible bias by asking respondents to select a project and answer questions about the background of a project in the early sections of the questionnaire. The concrete questions, which may involve sensitive information, were put in later sections of the questionnaire. Thus, possible side effects of sensitive questions on the selection of projects were reduced.

Different people in the same project might have had different opinions about the same project, especially if some time has passed since the project was completed. Hence, asking only one person in each project might not reveal the whole picture. However, our samples were selected randomly. In many cases, we had no previous contact with the participants. Hence, it would have been almost impossible to get two independent people to answer questions on the same project.

Although all questionnaires were completed by using the Web-based tool in Norway and Italy, for Germany, most of the questionnaires were completed by the researcher during telephone interviews. Such different methods of completing the questionnaire may have biased the results of the study.

6.5.2 External Validity

Although we used different techniques to select samples in different countries, our study was the first survey on component-based software engineering (and, indeed, software engineering in general) to use stratified random sampling of IT companies in several countries. However, our study focused on OTS components that satisfy our definition of a component which we presented in Section 2. Different conclusions may be reached by studying projects that use complex and large OTS packages such as ERP, CRM, or content management system solutions.

We primarily present overall results from the aggregated data set over three countries. In addition, we analyzed possible diverging results using nationality as a grouping criterion. Through dummy-variable coding [54, p. 303], we transferred the categorical variable "country" into two binary variables v1 and v2 (v1 = 1 and v2 = 1 for Norway, v1 = 0 and v2 = 1 for Italy, and v1 = 0 and v2 = 0 for Germany). We used *hierarchical multiple regressions*, as shown in Section 3.4, to verify the effect of nationality on the risk and risk-management correlations. The results are summarized in Table 12 and show that most of our findings (shown in Table 10), except for the risk-reduction activity related to inefficient defect localization, are still valid when the country variable is controlled. A further ANOVA analysis [46, p. 324] on the variables "costly defect localization" and "additional learning effort" illustrates that German respondents devoted significantly more effort to learning and encountered significantly fewer occurrences of the costly debugging problem than was the case for Italian and Norwegian respondents. The results presented in Table 12 also illustrate that the country variables v1 and v2 covary with several risk items. This shows that some "cultural" differences between countries need to be considered regarding risk management. Therefore, the possibility of generalizing our conclusions may need to be verified country by country.

6.5.3 Construct Validity

In our study, most variables and alternatives were taken directly or with little modification from the existing literature. However, we may have failed to consider some risks and risk-reduction activities that were not mentioned in the literature. Due to the exploratory nature of this study, we did not measure the risk item and risk-reduction activities with subitems. A single-item bias may therefore threaten the construct validity of our results. However, the questionnaire was pretested using a paper version by 10 internal experts and eight industrial respondents before being published on the SESE tool [44]. About 15 percent of the questions were revised on the basis of the pretest results. Another possible threat is that only a self-reported questionnaire was used to collect data in our study. That may have introduced a monomethod bias [55].

6.5.4 Conclusion Validity

In the entire survey, we just observed what had happened, without being able to control any variables. We assumed that the a priori probabilities of typical risks are the same in most OTS component-based projects. We therefore examined only the relations between the performed risk-reduction activities and the actual incidences of problems that arose during the projects that corresponded to the risks. Given that different projects might have slightly different

a priori probabilities of typical risks, we cannot exclude the following possibilities:

- Integrators did not recognize a particular risk and did not take any risk-reduction action. However, the risk did not materialize.
- Integrators recognized a particular risk and took appropriate risk-reduction action. However, the risk materialized anyway.

To answer **RQ3**, we analyzed correlations between 11 risk reduction activities and 13 risk items, which include 143 comparisons. A possible threat to the conclusion validity of our study is the issues of multiple comparisons [56]. However, our study was designed as an exploratory study. The significant correlations that we found were used to formulate the hypotheses.

7 Conclusion and Future Work

OTS-based development brings both benefits and risks. Although several studies have investigated and proposed risk-reduction activities in OTS component-based development, few have validated and compared the effects of risk-reduction activities on the corresponding risk items. Our study explored the occurrences of typical risks in OTS-based projects and compared the effectiveness of the risk-reduction activities that were performed. The data was collected from 133 OTS component-based projects in Norway, Italy, and Germany. This study is probably the first international software engineering survey to use public census-type data to obtain a stratified random sample of national software companies.

The results show that two of the most frequent problems in OTS-based projects are the incorrect estimation of the integration effort and costly debugging. The least frequent problems are the possible negative effects of OTS components on the quality of the system as a whole. We discussed several promising risk-reduction activities, such as adding more learning effort in the planning phase, evaluating the quality of OTS component more thoroughly in the selection phase, integrating unfamiliar components first, and monitoring the support reputation of component providers. We also formulated five new hypotheses, using, as a basis, the effectiveness of these activities on certain risk items. In addition, we found that several context variables, such as the experience of the developer, the number of components, and the requirements of the system with respect to quality, affect the relationships between risk-reduction activities and the occurrence of certain risk items.

The results also show that there are still unexplained risk-related factors, in spite of "reusing" and consolidating almost all of the proposed risk-reduction activities from the literature. The weak correlations between individual risk items and risk-reduction activities and the total explained variances (R^2) of the risk items by these risk-reduction activities may appear to be disappointing (in spite of significant p-values). However, the ***unexpected*** results will serve as inspirations for further research to uncover any remaining risk factors and to understand their relation to and effect on OTS-based projects. For instance, the results indicate that the "cultural" and "human-social" (personal/team/management) factors might hold some secrets.

A more qualitative study (beyond the one in [17]) combined with our newly formulated hypotheses should be initiated. We therefore intend to examine several industrial projects by estimating the risk probabilities before the project starts, follow the execution of the project, and measure the occurrence of problems corresponding to the risks after the project has been completed. The purpose is to investigate the causal effects of selected risk-reduction activities on the occurrence of associated risks.

Appendix

The Data Collection Process

In Norway, we gathered a company list from the Norwegian Census Bureau (SSB) [43] that contained large (100 or more employees), medium (20-99 employees), and small (5-19 employees) IT companies. Using the number of employees as the criterion of size, we selected the 115 largest software-intensive companies (100 large IT companies and 15 IT departments in the largest three companies in five other sectors), 200 medium-sized software companies (20-99 employees), and 100 small companies (5-19 employees) as the original contact list (415 companies in all).

In Italy, we first identified 43,580 software companies from the Yellow Pages and compiled a numbered list. We then selected companies from this list at random by using a random number generator. We visited the Websites of these randomly selected companies to confirm that they really were software companies. We verified that 196 of the companies were software companies and included them in the original contact list.

In Germany, we composed a list of companies using names from an organization similar to the Norwegian Census Bureau and selected 476 of them in a manner similar to that in which we selected companies in Norway. We then used an existing client database to obtain the contact information of these companies.

To avoid a bias in the process of data collection, we constructed the following guidelines for contacting potential respondents before the survey started: We first contacted them by telephone. If the potential respondents had undertaken suitable OTS-based projects and wished to participate in our study, we sent them a username and password for the Web-based SESE survey tool [44] and an electronic version of the questionnaire in MS Word format. The respondents could use either the SESE Web tool or the electronic version to complete the questionnaire. We also registered those potential respondents who did not want to complete the questionnaire. We logged the main reasons for not wanting to respond, such as "no software development," "no OTS-based projects," and "busy." In total, we contacted (by phone) 1,087 companies in three countries. Four hundred twenty-five (39 percent) of these 1,087 companies were software companies and had experience with OTS component-based development. Of the 425, 234 (55 percent) companies declared a willingness to participate in the study and the remaining 191 companies claimed that they were too busy to participate. We sent the questionnaire to these 234 companies. However,

we managed to collect data from only 127 (54 percent) companies. The remaining 107 companies, which claimed willingness without answering the questionnaire, mainly claimed that they were too busy.

ACKNOWLEDGMENTS

The authors would like to thank all of the participants for filling in the questionnaire. The authors are also grateful to the anonymous referees for their valuable comments. They would also like to thank Chris Wright for proofreading the paper.

REFERENCES

[1] C. Szyperski, D. Gruntz, and S. Murer, *Component Software—Beyond Object-Oriented Programming.* Addison-Wesley, 2002.
[2] L.C. Rose, "Risk Management of COTS Based System Development," *Component-Based Software Quality—Methods and Techniques,* pp. 352-373, Springer, 2003.
[3] B.W. Boehm, D. Port, Y. Yang, and J. Bhuta, "Not All CBS Are Created Equally: COTS-Intensive Project Types," *Proc. Second Int'l Conf. COTS-Based Software Systems,* pp. 36-50, 2003.
[4] J. Voas, "COTS Software—The Economical Choice?" *IEEE Software,* vol. 15, no. 2, pp. 16-19, Mar./Apr. 1998.
[5] J. Voas, "The Challenges of Using COTS Software in Component-Based Development," *Computer,* vol. 31, no. 6, pp. 44-45, June 1998.
[6] C. Abts, B.W. Boehm, and E.B. Clark, "COCOTS: A COTS Software Integration Lifecycle Cost Model—Model Overview and Preliminary Data Collection Findings," Technical Report USC-CSE-2000-501, USC Center for Software Eng., http://sunset.usc.edu/publications/TECHRPTS/2000/usccse2000-501/usccse2000-501.pdf, 2000.
[7] G. Kotonya and A. Rashid, "A Strategy for Managing Risk in Component-Based Software Development," *Proc. 27th EUROMICRO Conf.,* pp. 12-21, 2001.
[8] COTS Risk Factor, http://www.faa.gov/aua/resources/cots/Guide/CRMG.htm, 2003.
[9] B. Fitzgerald, "A Critical Look at Open Source," *Computer,* vol. 37, no. 7, pp. 92-94, July 2004.
[10] G. Lawton, "Open Source Security: Opportunity or Oxymoron?" *Computer,* vol. 35, no. 3, pp. 18-21, Mar. 2002.
[11] P. Vitharana, "Risks and Challenges of Component-Based Software Development," *Comm. ACM,* vol. 46, no. 8, pp. 67-72, Aug. 2003.
[12] M. Ruffin and C. Ebert, "Using Open Source Software in Product Development: A Primer," *IEEE Software,* vol. 21, no. 1, pp. 82-86, Jan./Feb. 2004.
[13] M. Torchiano and M. Morisio, "Overlooked Facts on COTS-Based Development," *IEEE Software,* vol. 21, no. 2, pp. 88-93, Mar./Apr. 2004.
[14] J. Li, F.O. Bjørnson, R. Conradi, and V.B. Kampenes, "An Empirical Study of Variations in COTS-Based Software Development Processes in Norwegian IT Industry," *J. Empirical Software Eng.,* vol. 11, no. 3, pp. 433-461, Sept. 2006.
[15] B.W. Boehm, "Software Risk Management: Principles and Practices," *IEEE Software,* vol. 8, no. 1, pp. 32-41, Jan. 1991.
[16] J. Li, M. Torchiano, R. Conradi, O.P.N. Slyngstad, and C. Bunse, "A State-of-the-Practice Survey of the Off-the-Shelf Component-Based Development Processes," *Proc. Ninth Int'l Conf. Software Reuse,* pp. 16-28, June 2006.
[17] J. Li, R. Conradi, O.P.N. Slyngstad, C. Bunse, M. Torchiano, and M. Morisio, "An Empirical Study on the Decision Making Process in Off-the-Shelf Component Based Development," *Proc. 28th Int'l Conf. Software Eng.,* pp. 897-900, May 2006.
[18] J. Li, R. Conradi, C. Bunse, M. Torchiano, O.P.N. Slyngstad, and M. Morisio, "Development with Off-the-Shelf Components: 10 Facts," *IEEE Software,* to be published.
[19] B.W. Boehm, *Software Risk Management,* tutorial, IEEE CS Press, 1989.
[20] H. Barki, S. Rivard, and J. Talbot, "Toward an Assessment of Software Development Risk," *J. Management Information Technology,* vol. 22, no. 2, pp. 359-371, Dec. 1993.
[21] M. Carr, S. Kondra, I. Monarch, F. Ulrich, and C. Walker, "Taxonomy-Based Risk Identification," Technical Report SEI-93-TR-006, SEI, 1993.
[22] S.A. Sherer, "The Three Dimensions of Software Risk: Technical, Organizational, and Environmental," *Proc. 28th Hawaii Int'l Conf. System Sciences,* pp. 369-378, 1995.
[23] C.G. Chittister and Y.Y. Haimes, "System Integration via Software Risk Management," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 26, no. 5, pp. 521-532, Sept. 1996.
[24] J. Ropponen and K. Lyytinen, "Components of Software Development Risk: How to Address Them? A Project Manager Survey," *IEEE Trans. Software Eng.,* vol. 26, no. 2, pp. 98-112, Feb. 2000.
[25] M. Keil, P.E. Cule, K. Lyytinen, and R.C. Schmidt, "A Framework for Identifying Software Project Risks," *Comm. ACM,* vol. 4, no. 11, pp. 76-83, Nov. 1998.
[26] L. Wallace and M. Keil, "Software Project Risks and Their Effect on Outcomes," *Comm. ACM,* vol. 47, no. 4, pp. 68-73, Apr. 2004.
[27] B.W. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer,* vol. 21, no. 5, pp. 61-72, May 1988.
[28] A. Gemmer, "Risk Management: Moving beyond Process," *Computer,* vol. 30, no. 5, pp. 33-41, May 1997.
[29] H. Hecht, *Systems Reliability and Failure Prevention.* Artech House, 2003.
[30] I. Sommerville, *Software Engineering,* seventh ed. Addison-Wesley, 2004.
[31] T. Moynihan, "How Experienced Project Managers Assess Risk," *IEEE Software,* vol. 14, no. 3, May/June 1997.
[32] R.L. Glass, "Frequently Forgotten Fundamental Facts about Software Engineering," *IEEE Software,* vol. 18, no. 3, pp. 110-112, May/June 2001.
[33] I. Crnkovic and M. Larsson, *Building Reliable Component-Based Software Systems.* Artech House, 2002.
[34] C. Szyperski, *Component Software: Beyond Object-Oriented Programming,* second ed. Addison-Wesley, 2002.
[35] M. Vigder, M. Gentleman, and J. Dean, "COTS Software Integration: State of the Art," Technical Report NRC No. 39190, 1996.
[36] V.R. Basili, M. Lindvall, I. Rus, C. Seaman, and B.W. Boehm, "Lessons-Learned Repository for COTS-Based SW Development," *Software Technology Newsletter,* vol. 5, no. 3, pp. 4-7, http://fc-md.umd.edu/ll/index.asp, 2002.
[37] T.R. Madanmohan and R. De', "Open Source Reuse in Commercial Firms," *IEEE Software,* vol. 21, no. 1, pp. 62-69, Jan./Feb. 2004.
[38] L. Bass, C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord, and K. Wallnau, "Volume I: Market Assessment of Component-Based Software Engineering," Technical Report CMU/SEI-2001-TN-007, SEI, http://www.sei.cmu.edu/, 2001.
[39] "Component-Based Software Engineering Network," CBSEnet, http://www.cbsenet.org, 2004.
[40] C. Abts, B.W. Boehm, and E.B. Clark, "COCOTS: A COTS Software Integration Cost Model—Model Overview and Preliminary Data Findings," *Proc. 11th European Software Control and Metrics Conf.,* pp. 325-333, Apr. 2000.
[41] R. Likert, "A Technique for the Measurement of Attitudes," *Archives of Psychology,* no. 140, pp. 5-55, 1932.
[42] J.R. Donald, V. Basili, B. Boehm, and B. Clark, "Eight Lessons Learned during COTS-Based Systems Maintenance," *IEEE Software,* vol. 20, no. 5, pp. 94-96, Sept./Oct. 2003.
[43] Norwegian Census Bureau (SSB), Oslo, ICT Company Data, http://www.ssb.no/emner/10/03/ikt/, 2002.
[44] E. Arisholm, D.I.K. Sjøberg, G.J. Carelius, and Y. Lindsjørn, "SESE—An Experiment Support Environment for Evaluating Software Engineering Technologies," *Proc. 10th Nordic Workshop Programming and Software Development Tools and Techniques,* pp. 81-98, Aug. 2002.
[45] R. Conradi, J. Li, O.P.N. Slyngstad, C. Bunse, M. Torchiano, and M. Morisio, "Reflections on Conducting an International CBSE Survey in ICT Industry," *Proc. Fourth IEEE Int'l Symp. Empirical Software Eng.,* pp. 214-223, 2005.
[46] P. Spector, "Ratings of Equal and Unequal Response Choice Intervals," *J. Social Psychology,* vol. 112, pp. 115-119, 1980.
[47] P.F. Velleman and L. Wilkinson, "Nominal, Ordinal, Interval, and Ratio Typologies Are Misleading," *J. Am. Statistician,* vol. 47, no. 1, pp. 65-72, Feb. 1993.
[48] D.J. Hand, "Statistics and Theory of Measurement," *J. Royal Statistical Soc.: Series A (Statistics in Soc.),* vol. 159, no. 3, pp. 445-492, 1996.

[49] B.H. Cohen, *Explaining Psychological Statistics,* second ed. John Wiley & Sons, 2000.
[50] *Common Risks and Risk Reduction Actions for a COTS-Based System,* www.mitre.org/work/sepo/toolkits/risk/taxonomies/files/CommonRisksCOTS.doc, 2005.
[51] V.R. Basili and B.W. Boehm, "COTS-Based Systems Top 10 List," *Computer,* vol. 34, no. 5, pp. 91-93, May 2001.
[52] S. Lauesen, "COTS Tenders and Integration Requirements," *J. Requirements Eng.,* vol. 11, no. 2, pp. 111-122, 2006.
[53] J. Li, R. Conradi, O.P.N. Slyngstad, C. Bunse, U. Khan, M. Torchiano, and M. Morisio, "Validation of New Theses on Off-the-Shelf Component Based Development," *Proc. 11th IEEE Int'l Software Metrics Symp.,* (abstract), p. 26, Sept. 2005.
[54] J. Cohen, P. Cohen, S.G. West, and L.S. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences,* third ed. Lawrence Erlbaum Assoc., 2002.
[55] P.M. Podsakoof and D.W. Organ, "Self-Reports in Organizational Research: Problems and Prospects," *J. Management,* vol. 12, no. 4, pp. 531-544, 1986.
[56] S.J. Pocock, M.D. Hughes, and R.J. Lee, "Statistical Problems in Reporting of Clinical Trials: A Survey of Three Medical Journals," *The New England J. Medicine,* vol. 317, no. 7, pp. 426-432, 1987.

Jingyue Li received the PhD degree in computer science from the Norwegian University of Science and Technology (NTNU) in 2006. He is presently a research scientist at NTNU. He worked at IBM China Ltd., before he came to NTNU. His research interests include software engineering, commercial-off-the-shelf-based development, open source-based development, project risk management, and aspect-oriented programming. He is a member of the IEEE Computer Society and the ACM.

Reidar Conradi received the MS and PhD degrees from the Norwegian University of Science and Technology (NTNU), Trondheim, in 1970 and 1976, respectively. He has been at NTNU since 1975. He is now a professor in the Department of Computer and Information Science (IDI), NTNU. He was a visiting scientist at the Fraunhofer Center for Experimental Software Engineering in Maryland and at the Politecnico di Milano in 1999 and 2000. His current research interests lie in software quality, software process improvement, version models, software evolution, open source software, and associated empirical studies. He is a member of the IEEE.

Odd Petter N. Slyngstad received the MSc degree in computer engineering from Iowa State University in 2002. He is currently a research fellow (PhD candidate) with the Norwegian University of Science and Technology, Trondheim, Norway. His research interests include risk management, software architecture, software evolution, and components off-the-shelf/open source systems. He is a student member of the IEEE and the Tekna.

Marco Torchiano received the MSc and PhD degrees in computer engineering from the Politecnico di Torino, Italy, where he is currently an assistant professor. He was a postdoctoral research fellow at the Norwegian University of Science and Technology (NTNU), Norway. He has published more than 50 research papers in international journals and conferences. He is a coauthor of the book *Software Development—-Case Studies in Java* (Addison-Wesley) and is a coeditor of the book *Developing Services for the Wireless Internet* (Springer). His current research interests include empirical software engineering, OTS-based development, and software engineering for mobile and wireless applications. He is a member of the IEEE Computer Society.

Maurizio Morisio received the PhD degree in software engineering and the MSc degree in electronic engineering from the Politecnico di Torino, Turin, Italy, where he is currently an associate professor. He spent two years working with the Experimental Software Engineering Group at the University of Maryland, College Park. His current research interests include experimental software engineering, service engineering, software reuse metrics and models, agile methodologies, and commercial off-the-shelf processes and integration. He is a consultant for improving software production through technology and processes. He is a member of the IEEE Computer Society.

Christian Bunse received the degree in computer science with a minor in medicine from the Universities of Bochum and Dortmund in Germany in 1993 and the PhD degree in computer science from the Technical University of Kaiserslautern, Germany, in 2000. Currently, he is an associate professor of software engineering in the School of Information Technology at the International University in Germany, Bruchsal. His research interests are in software methodologies, modeling, component and service orientation, resource awareness, software reuse, and empirical software engineering. The results of his research are documented in numerous publications in international journals and conference proceedings. In addition, he is a coauthor of several books on component and model-based development and software process models published by Addison-Wesley, Springer, and Elsevier. He is a member of the German Computer Society and the special interest groups in software technology, software architecture, and automotive software engineering.