Empir Software Eng (2006) 11: 433–461
DOI 10.1007/s10664-006-9005-5

An empirical study of variations in COTS-based software development processes in the Norwegian IT industry

Jingyue Li · Finn Olav Bjørnson · Reidar Conradi · Vigdis B. Kampenes

Published online: 24 May 2006

Editors: Mark Harman, Bogdan Korel, Panos Linos, Audris Mockus, and Martin Shepperd

Abstract More and more software projects use Commercial-Off-The-Shelf (COTS) components. Although previous studies have proposed specific COTS-based development processes, there are few empirical studies that investigate how to use and customize COTS-based development processes for different project contexts. This paper describes an exploratory study of state-of-the-practice of COTS-based development processes. Sixteen software projects in the Norwegian IT companies have been studied by structured interviews. The results are that COTS-specific activities can be successfully incorporated in most traditional development processes (such as waterfall or prototyping), given proper guidelines to reduce risks and provide specific assistance. We have identified four COTS-specific activities—the build vs. buy decision, COTS component selection, learning and understanding COTS components, and COTS component integration – and one new role, that of a knowledge keeper. We have also found a special COTS component selection activity for unfamiliar components, combining Internet searches with hands-on trials. The process guidelines are expressed as scenarios, problems encountered, and examples of good practice. They can be used to customize the actual development processes, such as in which lifecycle phase to put the new activities into. Such customization crucially depends on the project context, such as previous familiarity with possible COTS components and flexibility of requirements.

J. Li (✉) · F. O. Bjørnson · R. Conradi
Department of Computer and Information Science,
Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
e-mail: jingyue@idi.ntnu.no

F. O. Bjørnson
email:bjornson@idi.ntnu.no

R. Conradi
e-mail: conradi@idi.ntnu.no

R. Conradi · V. B. Kampenes
Simula Research Laboratory, P.O.BOX 134, NO-1325 Lysaker, Norway

V. B. Kampenes
e-mail: vigdis@simula.no

Keywords Software process improvement · COTS-based development · Component-based development

1. Introduction

COTS-based development has become more and more important in software and system development. Using COTS components promises faster time-to-market and increased productivity (Voas, 1998a). At the same time, COTS-based software introduces many risks. This means that unknown quality properties of the chosen COTS components can be harmful for the final product, and a COTS vendor may terminate the maintenance support of its COTS components (Voas, 1998b).

The use of COTS components introduces new system issues, which require revised software development processes. Although researchers and practitioners have been grappling with these new processes, most studies have been based on military or aerospace projects (Morisio et al., 2000; SEI, 2004), or similar large projects. In order to propose and design cost-effective COTS-based development processes, it is necessary to empirically investigate how COTS-based projects were performed in different application domains, particularly in small or medium-sized companies.

This study has investigated the commonalities and differences between development processes in 16 finished COTS-based software projects in Norway. It also summarized process scenarios in successful vs. unsuccessful COTS-based projects. Eleven salient problems encountered and 12 examples of good practice are extracted. A process customization guideline is proposed to help software practitioners to integrate relevant COTS components activities successfully in different project contexts.

The remainder of the paper is structured as follows: Section 2 presents previous studies on COTS-based development processes. Section 3 describes the research approach. Contexts of selected projects are presented in Section 4. The results are presented in Section 5 and discussed in Section 6. Our conclusions and proposals for future research are presented in Section 7.

2. COTS-Based Software Development Processes

2.1. COTS Component Definition

A pertinent question is "What do you mean by a COTS component?" There are many different definitions of COTS components (Basili and Boehm, 2001; Carney and Long, 2001). We have used the definition by Torchiano and Morisio (2004), where a COTS component should satisfy all the following requirements:

- It is either provided by some other organizations in the same company, or provided by external companies as a commercial product.
- It is integrated into the final delivered system.
- It is not a commodity. It is not shipped with an operating system, not provided with the development environment, and not generally included in any pre-existing platforms.
- It is not controllable by the user, in terms of provided features and their evolution.
- It is "black box" software. The source code is not available.

The granularity of a COTS component can be different (Torchiano and Morisio, 2004). Some consider that COTS components could or should include very large software packages such as Microsoft Office. Others limit COTS components to GUI libraries. In this study, we focus on COTS components as *software components*. Such a component is a unit of composition, and must be specified so that it can be composed with other components and integrated into a system (product) in a predictable way (Crnkovic et al., 2002). That is, a component is an "*Executable unit of independent production, acquisition, and deployment that can be composed into a functioning system.*" This definition implies that we include not only components following COM, CORBA, and EJB standards, but also software libraries like those in C++ or Java. This definition is consistent with the scope used in the component marketplace (Componentsource, 2004).

2.2. COTS-Based Development Processes

Typically, a COTS-based process consists of four phases, comprising (Abts et al., 2000):

- COTS component assessment and selection
- COTS component tailoring
- COTS component integration
- Maintenance of COTS and non-COTS parts of the system

There is consensus that the use of a COTS component implies changes in the software process (Brownsword et al., 2000). Most studies of a COTS-based development process focus on two dimensions: process of the whole software development lifecycle and process of the specific phase, especially in COTS component selection and evaluation.

2.2.1. Process of the Whole Software Development Lifecycle

Boehm and Abts (1999) regard both the waterfall model and evolutionary development as unsuitable for COTS-based development because:

- In the sequential waterfall model, requirements are identified at an earlier stage and the COTS components are chosen at a later stage. This increases the likelihood of the COTS components not offering the required features.
- Evolutionary development assumes that additional features can be added if required. However, COTS components cannot be upgraded for one particular development team. The absence of source code bars the development team from adjusting the COTS components to their needs.

Based on the above arguments, Boehm and Abts (1999) proposed that development models which explicitly take risk into account are more suitable for COTS-based development than the traditional waterfall or evolutionary approaches.

The Software Engineering Institute has a large ongoing effort to address the development of COTS-based systems, and they have developed the Evolutionary Process for Integrating COTS-based Systems (EPIC) (Albert and Brownsword, 2002). The proposed process integrates COTS-related roles and activities into a RUP process. The iterative and evolutionary nature inherent in this process allows developers to adjust the architecture and system design, as more knowledge is gained about the operations of the COTS components.

The National Aeronautic and Space Administration (NASA) has been developing IT systems with COTS components for many years. Their experience has been captured by Morisio et al. (2000). They have investigated the various processes that were used across 15 projects at NASA. Based on their insight, they have developed a COTS-based development process that was the most representative for the processes used in the projects.

2.2.2. COTS Component Selection and Evaluation Process

Based on case studies, researchers have proposed several COTS component selection processes and methods. One kind of process is based on direct assessment (Leung and Leung, 2002), such as MBASE (Model Based Architecting and Software Engineering) (Boehm, 2000), OTSO (Off-the-Shelf-Option) (Kontio, 1996), CISD (COTS-based Integrated System Development) (Tran et al., 1997), PORE (Procurement-Oriented Requirement Engineering) (Maiden and Ncube, 1998), CAP (COTS Acquisition Process) (Ochs et al., 2001), IIDA (Infrastructure Incremental Development Approach)(Fox et al., 1997), CARE (COTS-Aware Requirements Engineering) (Chung and Cooper, 2002), RCPEP (Lawlis et al., 2001), and CRE (COTS-Based Requirements Engineering) (Alves and Fineklstein, 2003). In general, the direct assessment process includes three basic steps:

- Inspect all modules of each of the available COTS components to check whether they satisfy some or all of the functional requirements of the COTS-based system being developed.
- Check whether a COTS component also satisfies the non-functional requirements of the COTS-based system. Non-functional requirements may include properties such as the interoperability of the modules of the COTS product with other systems.
- Compare the COTS component candidates and select the most appropriate COTS component that satisfies both the functional and non-functional requirement of the COTS-based system.

Some of the direct assessment processes, such as OTSO (Kontio, 1996), CAP (Ochs et al., 2001), and CISD (Tran et al., 1997), assume that the requirements are fixed and select the COTS components by comparing how well the COTS component candidates satisfy the requirements. A formal decision-making process is usually used to select the best COTS component (Ncube and Dean, 2002). The formal decision-making process usually includes three basic elements: selecting evaluation criteria (factors), collecting and assigning values to these criteria, and applying formal decision-making algorithms such as MAUT (MacCrimmon, 1973), MCDA (Morisio and Tsoukias, 1997), and AHP (Saaty, 1990). Other direct assessment processes, such as MBASE (Boehm, 2000), PORE (Maiden and Ncube, 1998), IIDA (Fox et al., 1997), CARE (Chung and Cooper, 2002), RCPEP (Lawlis et al., 2001), and CRE (Alves and Fineklstein, 2003) emphasize the trade-offs between the requirements and COTS component functionalities. These processes do not assume that the requirements are fixed and should not be changed. On the other hand, these processes propose that the requirements in COTS-based development should be flexible enough so that they can be negotiated or changed according to the available functionalities of the COTS components.

Another kind of process is based on a domain-model (Leung and Leung, 2002). It includes a set-up phase and a selection phase. In the set-up phase, the vendors need

to map their COTS modules to those modules of the domain that they find are applicable. In the selection phase, the corresponding modules in a domain model are identified for each of the modules of the COTS-based system in question. Then, the COTS modules that claim to be applicable are identified by the mapping from the domain models to the COTS modules. After that, the non-functional properties of the identified COTS modules are assessed. In the end, the most appropriate COTS modules are selected with reference to all of the assessment results.

3. Research Approach

Most studies on COTS-based development processes are limited to proposals of new technology (e.g., revised processes) without empirical evidence, or to discovery of existing practice (e.g., component selection) from case studies. Empirical studies are few and mostly based on a small project sample or on large projects with rather similar contexts, such as from aerospace or defense. It is therefore difficult to identify the precise relationship between a *project context* and COTS-based development processes. It is consequently hard for project managers to customize their COTS-based processes according to their project context.

Our research was designed as an exploratory study, i.e., mostly a qualitative study. The focus is to investigate the variations in COTS-based development processes and summarize these variations by scenarios. The intent is to find out how to customize a COTS-based development process based on its project context. Our research focuses on the development phases, not on software maintenance and evolution.

3.1. Research Questions

3.1.1. Research Question RQ1

Boehm and Abts (1999) regard both the waterfall and evolutionary lifecycle process as unsuitable for COTS-based development. Most COTS-based projects must therefore adjust their development process to a risk-driven spiral process, instead of a waterfall and evolutionary one. However, there are still no conclusive empirical studies on this issue. So, our first research question is:

*RQ*1: *What was the actual process used in the projects using COTS components?*

3.1.2. Research Question RQ2

Morisio et al. (2000) have proposed that COTS-based software processes differ considerably from "traditional" lifecycle processes, such as waterfall, or prototyping. Not modifying the traditional process can be a failure factor. They also pointed out that their proposed process has several variations for customization purposes. They also indicated that it is important to investigate under what conditions each variation is most suitable. However, there have been no further studies on this so far. Our second research question is then:

*RQ*2: *What are the commonalities and possible variations in COTS-based development processes?*

3.1.3. Research Question RQ3

All previous studies on COTS-based development processes have mentioned the possible problems (or risks) in using COTS components (Boehm and Abts, 1999; Morisio et al., 2000; Albert and Brownsword, 2002). To successfully customize COTS-based processes, it is important to summarize and understand the experience from successful COTS-based projects vs. those that were not so successful. It is especially valuable to generalize the relationship between possible risks and the process adjustments (variations). It may help project managers to successfully adopt a COTS-based development process without taking unnecessary risks. So, our third research question is:

> *RQ*3. *What are process scenarios (variations) of the projects using COTS components successfully and not successfully? What are possible problems and good practices in different process scenarios?*

3.2. Data Collection Method

The initial plan for this study was to make a quantitative survey using a pre-stated hypothesis with standardized questions in a formal questionnaire. After two rounds of pre-tests on the questionnaire, we found that the questionnaire requires substantial work to give consistent and reliable results. We decided, as a first step, to go for an exploratory pre-study using semi-structured interview.

Face-to-face interviews offer the possibility of modifying one's line of enquiry, following up interesting responses and investigating underlying motives in a way that questionnaire cannot (Robson, 2002). The choice of the semi-structured interview approach enabled more open research questions, not fixed hypotheses, but still relying on a rather detailed interview guide with both closed and open-ended questions. The advantages of open ended questions are that they allow us to go into depth and clear up misunderstandings, enable testing of limits of a respondent's knowledge, allow us to make a truer assessment of what the respondents really believe, and opening for unexpected or unanticipated answers.

3.3. Interview Guide

We used a modified version of the original questionnaire to serve as an interview guide. The interview guide includes both closed and open-ended questions.

The closed questions were used to solicit information on project context. For example, one question used the application domains of the final system (product), which was based on the North American Industrial Classification System (NAICS) (NAICS, 2004).

The open-ended questions were used to gather information on actual COTS-based development processes. The questions covered the main lifecycle process, the process changes due to using COTS components, and the experience and lessons learned in the whole COTS-based development process. Some of the questions were arranged in an "if-then" structure that leads the interviewer along one of several paths depending on the answers to previous questions. For example, one question

was to ask about whether they had changed or added some activities in their process because of using a COTS component. We then asked when and how did they make these changes if they added or changed some activities.

The interview guide was written in English. The reason was that we planned to extend this study to other countries later. On the first page, we gave the definitions of most concepts that were used in later questions, such as COTS component, project, and system (consisting of application, components, and addware/glueware). The rest of the interview guide had questions organized in three sections:

- Questions to characterize companies, projects, and COTS components.
- Questions to characterize the individual respondents.
- Questions on COTS-based development processes, and covering mainly COTS component selection and integration.

To evaluate the construct validity of the interview guide, we added three meta-questions following each question. These meta-questions asked the respondents, if the question was clear, if respondents needed clarification, and if respondents could give specific comments if the question was confusing. All this is a standard procedure from social science (Fowler, 2001).

Lastly, to ensure that a respondent had understood the COTS-relevant definitions correctly and could select a proper company project, we asked the respondent to give further examples of COTS components based on their understanding.

3.4. Sampling

Interviews and data collection were conducted by Li and Bjørnson from Nov. 2003 to Jan. 2004. They contacted 34 IT companies, where most were involved in two Norwegian R&D Projects (INCO, 2000; SPIKE, 2002) and one EU R&D project (FAMILIES, 2003).

A pre-process was used to select relevant COTS-based projects. Contact persons (IT manager in most cases) in the above projects were first contacted by phone and email, using an introductory letter with our definition on COTS component and project (the same as definitions in the interview guide). They were asked to check if they have a relevant COTS-based project. During this pre-process, 17 companies (among 34) did not match our definition of a COTS-based project, and therefore could not join the study. Of these 17 "rejected" companies, 13 used only in-house built components and three used only open source components. There was one other company that could not participate because of confidentiality issues.

For the remaining 17 companies, 13 of these volunteered to join in the study. The contact persons in these 13 companies were asked to recommend persons with most experience on the relevant projects. In all, 16 persons were recommended from these 13 companies. For each recommended respondents, they were asked to select one relevant COTS-project. If a respondent claimed that he/she had experience from several COTS-based projects, the respondent was asked to select one that he/she was most familiar with. The whole process to find a suitable respondent and project in a prospective company, and later agree upon place and time for a physical interview took weeks.

Although the total sample was based on convenience, we ensured that we had included respondents from large, medium and small companies.

3.5. Data Collection Procedure

The interview guide was sent to respondents a few days before the personal interview. In this way, the respondents were well-prepared and could have time to remember the details of the projects and find some relevant documents.

We offered the company respondent a compensation of 100 euro and/or a copy of a status report. Most companies did not want money, rather a status report. We emphasized that all answers and other information would be treated strictly as confidential.

Most of the answers and comments requested by the interview guide were filled-in on paper during the interview by the interviewer. Each interview was conducted by one of the two interviewers (none with two interviewers), took from 60 to 80 min, and was recorded.

3.6. Data Analysis

The first step in data analysis was to listen to the tape, and supplement and transcribe the interview "as-is" into a field note. After that, we used different analysis procedures according to the purpose of different research questions.

The purpose of the first *research question RQ*1 is to investigate the actual development processes used in the project. The data was analyzed by the constant comparison method (Strauss and Corbin, 1998). As all respondents had a clear definition on the main process used in their project. These processes were early grouped into traditional process (waterfall, prototyping, or incremental) or a totally new process.

The purpose of the second *research question RQ*2 is to see the similarities and differences between process changes in the studied projects. The main analysis method used is cross-case analysis (Strauss and Corbin, 1998), i.e., treating each project as a separate "case." The field notes for the first two projects were first reviewed. For each of these two projects, a list was compiled with short phases that describe the process changes in each project (i.e., new or changed activities, when and how these changes were performed). Then these two lists were compared to determine the similarities and differences. The next step was to list, in the form of propositions, conclusions one would draw if these two projects were the only two in the data set. Each proposition had associated process changes that supported it. After analyzing the first two projects in this way, the third project was examined, and a list of its process changes was compiled. Then it was determined whether this third project supported or refuted any of the propositions formulated from the first two. If a proposition was supported, this third project was added to its list of supporting evidence. If it contradicted a proposition, then either the proposition was modified or the project was noted as refuting that proposition. Any additional propositions suggested by the third inspection were added to the list. This process was repeated for each project. The end result was a list of propositions, each with a set of supporting and refuting evidence (projects).

The purpose of the third *research question RQ*3 is to summarize the process scenarios of the successful and unsuccessful projects. We first grouped projects into two groups based on the positive or negative effect of the COTS components regarding time-to-market and general quality of the system. In each group, we divided projects into sub-groups based on their main actual processes used. For each

sub-group, we performed a similar cross-case analysis as for research question *RQ2*. The difference is that we included more information for analysis. For each project, a list of short phases was compiled that describe not only the process changes in each project, i.e., new or changed activities, but also when and how these changes were performed. We also recorded project contexts, i.e., the developers' experience with COTS components. To summarize the lesson learned in different process scenarios, we selected the main problems met and examples of good practice performed in each process scenario. The problems and examples of good practice that occurred in more than one project were grouped into one item. Some problems and examples of good practice, which occurred in only one project, were also listed because they were regarded as the main factor that decided the success and failure of the project.

4. Collected Samples

4.1. Companies

As mentioned, we interviewed 16 projects in 13 companies. All the companies are Norwegian IT companies. Nine of these 13 are stand-alone companies, with staff size between 5 and 500. The other companies are subsidiaries, with local staff size ranging from 50 to 320. Six companies are IT consultancies, five are software vendors, and two are telecom companies. Five of these companies are publicly traded companies, the others are privately held. Some company background information is listed in Table 1.

4.2. Respondents

We had one respondent from each project. Three respondents are IT managers, four are project managers, five are software architects, three are software developers, and one is a software development researcher. Nine of them have more than 10 years of software development experience, and 12 of them have more than 4 years of working experience with COTS-based development. Eight of them have a master

Table 1 Background information on the 13 IT companies (called C1 to C13)

Company ID	Staff size	Type	Ownership	Main business area
C1	42	Stand-alone	Privately held	IT Consulting
C2	60	Subsidiary	Privately traded	IT Consulting
C3	36	Stand-alone	Privately held	IT Consulting
C4	17	Stand-alone	Privately held	IT Consulting
C5	500	Stand-alone	Publicly traded	IT Consulting
C6	320	Subsidiary	Publicly traded	Telecom Industry
C7	120	Stand-alone	Publicly traded	Software Vendor
C8	100	Stand-alone	Privately held	Software Vendor
C9	130	Stand-alone	Privately held	IT Consulting
C10	275	Subsidiary	Publicly traded	Telecom Industry
C11	50	Subsidiary	Publicly traded	Software Vendor
C12	5	Stand-alone	Privately held	Software Vendor
C13	80	Stand-alone	Privately traded	Software Vendor

degree, and the rest have a bachelor's degree. Twelve of them have a principal degree in informatics or computer science.

4.3. Projects

The company projects were selected based on two criteria:

- The project should use one or more COTS components
- The project should be a finished project and possibly with maintenance, and possibly with several releases.

Generally, we selected only one project from a company. However, we selected two projects in three companies because these projects differed largely in project members, development process, and COTS components used.

Background information includes effort spent, development language, and application domain. Table 2 lists the basic information on the 16 projects.

4.4. COTS Components

Some projects used one or two COTS components, others used more. Because of time limitations, we asked the respondents to select no more than three typical COTS components which were used in their projects. Totally, we gathered information about 30 different COTS components in 16 projects. Some typical COTS components are listed in Table 3. The list is not exhaustive, and is only used to indicate the kinds of selected COTS components. Some of the COTS components are based on standards like COM, CORBA, and EJB, and some of them are C++ or Java libraries. All the COTS components follow our previous definition, i.e., they were integrated into the final product and were delivered together with the final system to the customer.

Table 2 Background information on the 16 projects (called P1 to P16)

Company ID	Project ID	Effort (person-hours)	Main development language	Application domain
C1	P1	7,000	Java	Finance
C1	P2	30,000	Power Builder	Finance
C2	P3	700	C, Java	Scientific
C2	P4	42,000	C	Scientific
C3	P5	16,000	C++	Internet Service provider
C3	P6	23,000	C++	Telecom.
C4	P7	2,400	Java	Agriculture
C5	P8	3,000	Visual Basic	El. Power
C6	P9	58,000	C++, Java	Telecom
C7	P10	63,000	C++, VB	Whole trade
C8	P11	8,000	C++	El. Power
C9	P12	10,000	C++	Construction
C10	P13	128,000	C++	Telecom
C11	P14	120,000	C++	Transport
C12	P15	92,000	Ada, Java	Finance
C13	P16	15,000	C++	Scientific

Table 3 Background information on some COTS components

COTS component name	Functionality
CTI	Computer telephony integration
Sheridan	GUI
Intelligent Report	Report generator
Active Reports	Report preview and printing
Open MP	Open parallel processing
IMAPP	Image processing
SearchEngine	Searching the context in the website
ComWork	Workflow management
XML lite	XML parser

5. Results

5.1. Overall Process Used in COTS-Based Software Development

Research question *RQ*1 is to identify the actual processes used in projects with COTS components. To study this issue, we asked two open-ended questions: we first ask the respondents to describe the development processes they used. We also asked if they selected the actual development processes because of using COTS components.

Results of the first question show that the main development processes of the studied projects can be grouped into three categories: *pure waterfall, waterfall mixed with prototyping, and incremental mixed with prototyping*. Five projects used waterfall. One used waterfall mixed with little prototyping (a small prototype was used for discussing requirements of the GUI part of the system). Ten projects used incremental mixed with some prototyping.

Results of the second question show that none of the respondent claimed that they selected the main development process based on whether or not they intended to use COTS components. That is, they decided the main process before they started to think about using COTS components or not. They just added or changed some activities and roles in their traditional process to reflect the use of COTS component.

Therefore, the answer to research question *RQ*1: *The so-called COTS-based development process is the customization of the traditional development process due to the use of COTS components.*

5.2. Commonalities and Variances in COTS-Based Software Development Process

Research question *RQ*2 investigates the commonalities and possible variations in the development processes. To study the common changes in the develop processes, we asked if there are any new or changed activities, role and responsibilities in the project due to use of COTS components. The detailed answers are listed in Appendix A. The commonalities are that new activities were added and possible one new role was added. The added new activities include:

– *Build vs. buy decision*: In COTS-based development, the first extra activity in all the studied projects was to make the build vs. buy decision. Non-technical issues, such as costs, licensing, and market trends, as well as technical issues may dominate this stage.

- *COTS component selection*: after the respondents decided to use COTS component; the following extra activity in all the 16 projects was COTS component selection. However, different projects used different selection processes and criteria in their COTS component selection.
- *Learn and understand the COTS components*: another new activity is to learn and understand the COTS components. Developers needed to spend time to study several possible candidates briefly before the final selection. After they have selected the COTS components, they needed to spend time to learn how to use the selected COTS components. Because the source code of a COTS component is often not available, it is difficult for developers to profoundly understand a COTS component.
- *Build glueware and addware*: a future step after COTS component selection is integration. All the projects built more or less glueware (the code to integrate the COTS components) and/or addware (the code to complete the required functionality of COTS components).

The possible added new role is: a *COTS component knowledge keeper*. As mentioned above, it was difficult to really understand the COTS components. In ten projects, one or more project members had some previous experience with selection and integration of the actual COTS components. They provided suggestions on COTS component selection and integration. Although some of them were not dedicated to work as a COTS component knowledge keeper, their previous experience improved the speed and quality of COTS component selection and integration.

To investigate the variations in the processes, we transcribed respondents' description on the above new activities and compared when, how, and why these activities are performed. The detailed information on when and how the COTS components were selected is listed in Appendix B. The results show that the main *variations* in the customization of main development processes deal with the following activities:

- *The phase to make the build vs. buy decision and the COTS component selection*: Morisio et al. (2000) proposed a two-phase build vs. buy decision. The first decision is in the requirement phase and the second in the design phase. Our results indicate that only two of the five waterfall projects made the build vs. buy decision and COTS component selection in the requirement phase. If the project members had no relevant experience with specific COTS component, it was very hard to make that choice in the requirement phase. Two waterfall process projects managed to make the build vs. buy decision and COTS component selection in the requirement phase, because they were already familiar with the possible COTS components.
- *The COTS component selection and evaluation process*: the selection process in our studied projects can be summarized in two categories:

    1. *Familiarity-based selection process (SP_fam)*: the previous familiarity came either from the project members in the projects, i.e., the COTS component knowledge keeper, or from a colleague outside the project. If the project members had enough previous experience with the candidate COTS components, this experience will be the key factor in the COTS component selection. In all, 16 of the 30 given COTS components were selected based on the suggestion of the COTS component knowledge keeper inside the projects. Four COTS components were selected based on recommendations from colleagues outside the project or organization.

2. *Internet search, trial-based selection process (SP_unfam):* this kind of selection was used when there was no previous experience available. This selection was more complex than SP_fam, and involved three steps:

   Step 1: First, developers used a search engine to browse the Internet, using keywords to express decided functionality. This usually gave them a handful of candidates.

   Step 2: If there were more than 2 to 3 possible candidates, they selected 2 to 3 of them very quickly based on some key issues, such as license issues, cost, and vendor reputation.

   Step 3: After a small number of candidate COTS components had been decided, they downloaded a demo version of these from the web and tested the COTS components.

This kind of selection is similar to some proposed processes, such as OTSO (Kontio, 1996), CAP (Ochs et al., 2001), and CISD (Tran et al., 1997). The main difference is that the actual process was much faster and no decision-making algorithms have been used. In addition, developers knew that it was impossible to test several COTS components completely due to limited time and cost. They generally tested only some of the key functionalities, and depended on comments from a newsgroup to evaluate the quality of the COTS components. Therefore, COTS components with more and better comments in the newsgroup or marketplace Dashin had a good chance to be selected, because they were assumed to be better tested and in general of higher quality.

The answer to research question *RQ*2: *There are some common new activities and one new role is added in COTS-based development processes. The possible variations are when and how to perform these new activities.*

5.3. Scenarios of Projects that Used COTS Successfully and Unsuccessfully

Research question *RQ*2 is about summarizing the process scenarios of projects using COTS components successfully and unsuccessfully. The focus is to investigate the relationship between the possible risks and the development processes. We asked respondents to summarize their problems encountered (PEs) and their examples of good practices (GP) in the studied projects. First, we extracted process-relevant parts, i.e., what were the main development processes, and when and how were the new activities performed. We then summarized some other salient problems and examples of good practice, which may give guidance to COTS-based development.

In our definition, a project that used COTS components successfully means that the COTS components contributed positively to time-to-market and system quality. Otherwise, it is regarded as being used unsuccessfully.

5.3.1. Scenarios of Projects that Used COTS Components Unsuccessfully

Four of the 16 studied projects were regarded as using COTS components unsuccessfully. The scenarios are summarized in scenarios Sc1 to Sc3 in Table 4.

Table 4 The scenarios of the 4 projects that used COTS components unsuccessfully

Scenario ID (Project ID)	Characterizations	Problems encountered (PEs)	Good practice (GP)
Sc1 (P1 and P14)	Two projects used waterfall processes. Made the build vs. buy decision and the COTS component selection in the implementation phase. Selected unfamiliar COTS components.	**PE1**: Selected the COTS component too late. **PE2**: Used only demonstration, without pilot. **PE3**: Could not (re) negotiate requirements with customer. **PE4**: It was difficult to formally verify the COTS component's reliability.	**GP1**: Used external COTS expert.
Sc2 (P10)	One project used waterfall process. Selected COTS components in the design phase, but mainly on their functionality Selected unfamiliar COTS components.	**PE5**: Selection was based only on functionality. Ease of integration was not considered.	**GP2**: Used external COTS expert with internal developers.
Sc3(P2)	One project used incremental & prototyping process. Selected the COTS components in the design phase, but mainly because of their functionality. Selected unfamiliar COTS components.	**PE5**: Selection was based only on functionality. Ease of integration was not considered. **PE6**: COTS component's delay caused the delay of the project.	

The detailed backgrounds of each problem encountered are following:

- *PE*1: The two waterfall projects (P1 and P14) started to select COTS components in the implementation phase. It was very difficult for them to select the suitable COTS components, when the customer requirements and design have been fixed. They built a lot of glueware to integrate the selected COTS components, and this brought many integration problems at the very end.
- *PE*2: In project P1, the COTS component was selected only based on the vendor's demonstration. They did not do any hands-on trial to investigate how much effort is needed to integrate the COTS component into the current system. As mentioned in PE1, they had to spend a lot of effort on integrating the COTS components into the system, which caused delay in the project.
- *PE*3: Although some previous studies proposed to (re)negotiate requirements with the customer when developers found later limitations of COTS components, developers in project P14 could not (re)negotiate the requirements with customers even if they wanted to. This is because the developer wanted to use COTS components, not the customer. The customer cared only about the final result. If developers were not allowed to use COTS components, (re)negotiation of requirements due to COTS limitations is difficult.
- *PE*4: The final product of project P14 is a highly safety-critical system. To ensure reliability of the product, each line of code was required not only to be tested strictly, but also to be formally proved. As the source code of the COTS component was not available, they could not formally prove the correctness of the COTS component even the COTS component passed all existing testing. They were therefore always worried about it until they bought the source code.
- *PE*5: In project P10, the project members did not know exactly how to integrate the COTS components into the system and how to integrate them with other components before COTS component selection. Their customer intervened in the COTS component selection and recommended a certain COTS component that could provide the needed functionality. This COTS component caused a lot of quality problems.

Although these projects were regarded as being unsuccessful, there were still some examples of good practice that helped to improve the project results. The background of each example of good practices is the following:

- *GP*1: Project P1 hired one person from the vendor as a part-time consultant to configure and integrate the COTS. Although a lot of glueware was needed to integrate the COTS component because of the wrong selection, this external expert helped to shorten the integration time to some degree.
- *GP*2: In project P10, the whole component part was outsourced to a consulting company. The software company signed a fix-cost bid with the consulting company. The consulting company did all the work to make the integration work. As a result, the software company met the budget for the project. One of their architecture and project managers worked closely with consulting company. Therefore, these internal persons learned the COTS components in this project. They believed that they could save a lot of money in the following project, which will use the same COTS component.

5.3.2. Scenarios of Projects that Used COTS Component Successfully

Twelve projects successfully integrated the COTS components. The scenarios are summarized into scenarios Sc4 to Sc6 in Table 5.

The detailed explanations of each problem encountered are following:

- *PE*7: In project P4, they used a COTS component which includes more than 500 different classes. It took longer time to read the document and to understand how to use these classes than they originally estimated. As the COTS component provided most functionality, they still believed that it shortened the time-to-market of the system.
- *PE*8: In project P13, the project manager made a too optimistic estimation on the effort for the COTS component integration. Although they have used these COTS components before, it still took unexpectedly more time to integrate it into the new project. However, they still thought the COTS component provided positive effect on the time-to-market of the system.
- *PE*9: In project P13, they found some defects in the system during testing. However, it was difficult to convince the vendor that these defects were inside rather than outside the COTS components. The vendor took a long time to rectify these defects.
- *PE*10: In projects P3 and P9, developers found that the COTS components did not support the standard (such as EJB, DCOM) the vendor claimed. The COTS supported some old version standard instead of the new one as they claimed.
- *PE*11: In project P8, the COTS components could not work in the first round of integration testing. The COTS components failed because it could not work on the deployed platform. The reason is that developers used different platforms in the development and test.

To ensure the success of these projects, members of these projects performed some other activities to reduce some possible negative effects of COTS components. Their examples of good practices are summarized in the following:

- *GP*3: As the developers had experience with the relevant COTS components, they knew how to integrate them and their limitations. In the COTS components selection, they focused mainly on the ease of integration. If the COTS components cannot provide all the required functionality, they built some addware. In project P4, selecting COTS components based on the ease of integration was highly recommended.
- *GP*4: In project P4, they not only consulted an internal expert, but also followed the newsgroups of a COTS component on the Internet. Comments in the newsgroups helped them to make the right selection.
- *GP*5: In project P13, they investigated the architecture of the system and the COTS component to ensure the substitutability of the COTS component. This means that they can easily change to other COTS components, if they are not satisfied with the current COTS component. The same experience was also emphasized by respondents in projects P11 and P12.
- *GP*6: Several projects started to select COTS components in the requirements phase. This makes it simpler to match and possibly negotiate customers' (volatile) requirements with the available functionality of the COTS components, because the customer will accept the functionality of the COTS

Table 5 The scenarios of 12 projects that used COTS component successfully

Scenairo ID (Proj. ID)	Characterization	Problems encountered (PEs)	Good practice (GP)
Sc4 (P4 and P13)	Two projects used a waterfall process.	**PE7**: Under-estimated learning effort.	**GP3**: COTS component selection was based on architecture.
	Made the build vs. buy decision and selected COTS components in the requirement phase.	**PE8:** Under-estimated the integration effort.	**GP4:** Used COTS component newsgroup.
	Selected familiar COTS components.	**PE9:** It was difficult to isolate and prove bugs.	**GP5:** Prepared for replacement.
Sc5 (P12)	One project used waterfall with some prototyping process. Made the build vs. buy decision and selected COTS components in the design phase. Selected familiar COTS components.		**GP5:** Prepared for replacement.
Sc6 (P3, P5, P6, P7, P8, P9, P11, P15, and P16)	Nine projects used incremental with prototyping processes. Some projects selected COTS components in the requirements phase with familiar COTS components. Some projects selected COTS components in the implementation phase with familiar COTS components Some projects selected COTS component in the design phase with unfamiliar COTS components.	**PE10:** Met standard mismatch. **PE11:** Met deployment mismatch.	**GP3:** COTS component selection was based on architecture. **GP5:** Prepared for replacement. **GP6:** Using COTS components in the incremental & prototyping process helped the success of the project. **GP7:** Integrated unfamiliar COTS component first. **GP8:** Limited the amount of glueware. **GP9:** Did not change COTS code. **GP10**: Did integration testing incrementally. **GP11**: Paid vendor incrementally.

components shown in the prototype. At the start of the prototyping process, the main benefit of using COTS components is to build a prototype rapidly. They could also learn and understand how to use the COTS components in building such a prototype.

- *GP*7: In project P3, integrating unfamiliar COTS components was ranked as a high risk task. These unfamiliar components had been integrated and tested before other components.
- *GP*8: In project P3, one experience was that they tried to limit the amount of glueware. They believed that the less glueware they need to produce, the easier it would be to update in case a vendor provides new versions.
- *GP*9: In project P13, they had the source code of one COTS component. However, they decided not to change the code because if the vendor released a new version, it will be hard for the customer to maintain possible changes himself.
- *GP*10: In some projects that used more than one COTS component, the experience is that the COTS components should be integrated and tested incrementally. It means that the integration test should be performed immediately after one COTS component was integrated, instead of doing the final integration testing. Without incremental integration testing, it may be difficult to locate defects after all the COTS components are integrated.
- *GP*11: In project P7, the project paid part of the agreed amount after the evaluation. As the project members discovered some limitations of the COTS component, they asked the vendor to perform changes. After they got the component as required, they paid the rest of the money to the vendor. They believe that this allowed them to persuade the vendor to change the component.
- *GP*12: In project P15, they investigated the COTS component market share before they selected it. They believed the vendor can survive for a long time if their COTS products were used sufficiently in the market.

The studied projects showed that the COTS-based development could succeed under different process scenarios. The answer to research question *RQ*3: *COTS-based development could be performed successfully using waterfall and evolutionary processes. However, different process scenarios may face various risks.*

6. Discussion

6.1. Comparison with Related Work

Results of this study confirmed some conclusions from previous studies and contradicted others.

Boehm and Abts (1999) regard both the waterfall model and evolutionary development as unsuitable for COTS-based development. The results from research question *RQ*3 showed however that some projects could integrate COTS components successfully, using waterfall or evolutionary processes. However, the results of *RQ*3 revealed that different process scenarios (Sc1 to Sc6) may have different risks. It is therefore necessary to perform risk management according to customized process scenarios, as proposed by Boehm and Abts (1999). In addition, results of *RQ*3 showed that the main development processes were decided *before* the decision was made about whether to use COTS components. That is, if the process is risk-

driven as Boehm and Abts (1999) proposed, developers should use non-COTS related risks to decide the main development processes first. They should (only) start to consider COTS related risks, after they decide to use COTS components.

Both EPIC (Albert and Brownsword, 2002) and the process proposed by Morisio et al. (2000) indicated that some new activities and roles should be added to traditional development processes. Examples are the build vs. buy decision, COTS components selection, COTS components integration, and COTS component knowledge management. The results of *RQ2* gave further support to their conclusions. This study shows that there are several possible variations in COTS-based development process, for example, the COTS components were selected not only in the requirement phase, but also in the design or coding phases.

Although the formal decision-making algorithms, such as MAUT (MacCrimmon, 1973), MCDA (Morisio and Tsoukias, 1997), or AHP (Saaty, 1990), was used in some previous studies, such as OTSO (Kontio, 1996), CAP (Ochs et al., 2001), and CISD (Tran et al., 1997). The results of *RQ2* showed that none of the studied projects used these formal decision-making algorithms. The results support findings (named thesis T4) of Torchiano and Morisio (2004). The possible reason is that these techniques may not be applicable in selecting COTS (Ncube and Dean, 2002; Maiden et al., 2002). In practice, if developers had enough previous experience or tacit knowledge with specific components, COTS component selection was based on their experience (see SP_fam in Section 5.2). If they had no previous experience with the selected COTS components, the selection was based on hands-on trials (see SP_unfam in Section 5.2). A handful of candidate COTS components were compared very quickly with some key factors, for example price, platform, and vendor reputations. Two or three COTS components were selected for further hands-on trials.

Although methods to change or negotiate the requirements, such as MBASE (Boehm, 2000), PORE (Maiden and Ncube, 1998), IIDA (Fox et al., 1997), CARE (Chung and Cooper, 2002), RCPEP (Lawlis et al., 2001), and CRE (Alves and Fineklstein, 2003), have been proposed to select COTS products. The results of *RQ2* showed that these methods were not used in our studied projects. The possible reason is that most COTS components being used in our studied projects are fine-grained COTS components with limited functionalities (see Table 3). In case of using these fine-grained COTS components, it is difficult for developers to negotiate the requirements with the customer. As shown in PE3 (see Section 5.3.1), the customers can easily ask the developers to build the required functionalities in-house instead of using a COTS component, if the candidate COTS component cannot satisfy the customers' requirements.

COTS-based software introduces many risks (Voas, 1998b). Some previous studies summarized lessons learned and experiences by case studies (Boehm et al., 2003; COTS lessons learned repository, 2004; Tran et al., 1997; Fox et al., 1997; Rose, 2003; Morris et al., 2003; FAA Guide, 2003; Kotonya and Rashid, 2001). The results of this study gave further support to some of their conclusions. For example, faulty vendor claims may result in feature loss and/or significant delays (Boehm et al., 2003), and it is not recommend changing the source code of the COTS component even if such a code is available (COTS lessons learned repository, 2004; Morris et al., 2003). However, we discovered some new salient problems encountered and examples of good practice, such as incremental payment and preparation for replacement. In addition, previous studies did not investigate the relationship between the risks and corresponding

process scenarios. Their lessons learned are proposed as general guidelines. It is therefore difficult for practitioners to assess and evaluate the possible risks according to their project contexts. The results of our study illustrated that some risks occur more frequently in some development processes, such as waterfall process, than in other development processes, such as incremental process.

6.2. Customizing the COTS-Based Development Process

Our results show that the COTS-based development process is not a totally new process. It needs a customization (change or add new activities) on the traditional development processes. Based on the results of this study, we propose how to design and customize a COTS-based development process. As our studied projects mainly used waterfall and evolutionary processes, our proposal focuses on these. The process design could include two elements:

– First, decide the main development processes based on the project profile. The main development process could be decided based on risk considerations such as proposed by Boehm and Abts (1999), and based on the typical problems discovered in this study.
– If the main development process is waterfall or evolutionary, the process could be customized based on the actual main development process and project members' familiarity with relevant COTS components as following Fig. 1. For each customization, some possible risks must be identified and managed as we discovered from this study.

6.2.1. Waterfall with Unfamiliar COTS Components

Scenarios Sc1 and Sc2 showed that it is difficult to use an unfamiliar COTS component successfully if the general process is waterfall. It will be difficult to find COTS components to fully satisfy the requirements after the requirements and design have been decided upon. Therefore, our recommendation is that the build vs. buy decision and the COTS components selection should be performed before the implementation phase. The COTS components selection process could be

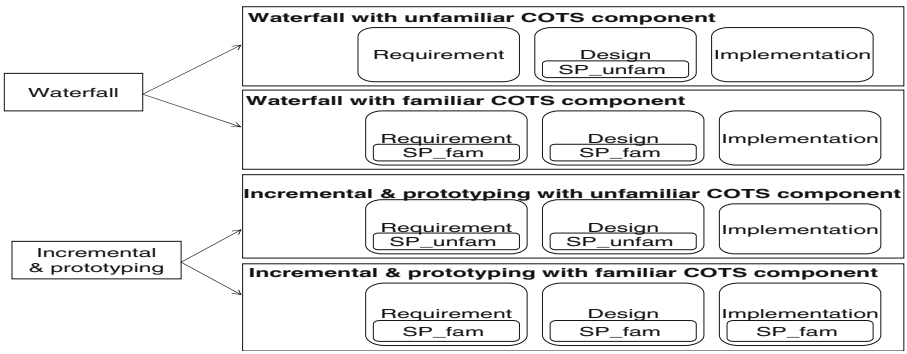

Fig. 1 Possible COTS-based development process customizations. SP_fam: Familiarity-based selection process. SP_unfam: Internet search, trial-based selection process

SP_unfam, and the ease of integration should be given much consideration in the build vs. buy decision. If the candidate COTS components do not appear to be easy to integrate, it is better to develop them in-house. Another recommendation is to inform the customer; if possible, about using COTS components after the build vs. buy decision. It may give the developers leeway to (re)negotiate the requirements if later limitations of COTS components are found.

6.2.2. Waterfall with Familiar COTS Components

If the project members are quite familiar with relevant COTS components, the build vs. buy decision and COTS component selection could be performed in the requirement phase, because the integrators know how to integrate them, and which functionality could be provided by the COTS components. Our results showed that COTS components selection drives the requirements to some extent.

6.2.3. Incremental & Prototyping with Unfamiliar COTS Components

Here, the risk of using unfamiliar COTS components is relatively low, compared to waterfall projects. The integrator can select the COTS components based on the completeness of required functionalities and build a prototype in a short time. The customers' requirements can be agreed upon by evaluating the prototype. For this customization, the recommendation is to integrate the unfamiliar COTS components first, if there are different phases to implement the system incrementally.

6.2.4. Incremental & Prototyping with Familiar COTS Components

If the project members have enough experience with relevant COTS components; they can easily build a prototype of the system with the COTS component. In this case, our results showed that buying COTS components had more advantages than building them in-house. Here, the build vs. buy decision can be performed in the requirements and/or design phase. If there are different phases to implement the system incrementally, they do not need to integrate all COTS components upfront. In this case, COTS components could be considered as in-house built components. The process could be the same as a non-COTS one.

6.3. Threats to Validity

6.3.1. Construct Validity

In this study, most variables are taken directly, or with little modification, from the existing literatures. To ensure construct validity, we did two rounds of pre-tests on the interview guide in ten local IT companies. These companies were selected based on convenience. For this, we wrote an introductory letter, sent by email. This was followed up by a phone call to see if the company had a willing contact person and a suitable COTS-based project. The interview guide (a MS-Word document) was then sent to the respondents by email. The respondents filled in the questions in the interview guide electronically, and gave free-text comments to each question and the following alternatives. Ten percent of the questions and alternatives in the interview guide were revised based on such comments.

The selected COTS component and further COTS component examples proposed from respondents show that the definition of a COTS component was understood correctly. Furthermore, the responses to the three meta-questions show that the questions in the interview guide were understood consistently.

However, one possible threat to construct validity is our chosen success criteria for a project successfully using COTS components, implying that the integrated COTS components should contribute positively both to time-to-market and to the quality of the final system. This may not apply for all kinds of projects as some projects might emphasis only time-to-market, and others might emphasis only quality.

6.3.2. Internal Validity

The first possible threat to internal validity is our misunderstanding of respondents' answers. Although two interviewers carried out the interviews and there was only one in each interview, we taped all interviews. Listening to the tape helped ensure correct interpretation of answers and comments. However, having an independent (third) person to listen to the tape might increase data quality.

The second possible threat to internal validity is that people tend to talk about examples of good practices instead of problems they encountered. As we had a confidentiality agreement with the respondents, the respondents were encouraged to tell the truth. The data in Tables 4 and 5 showed that 11 (out of 16) respondents stated at least one problem encountered in their projects. However, the respondents might give only the most salient problems they encountered and forget some minor problems because we asked for data about past projects.

The third possible threat to the internal validity is that the lifecycle phases, such as requirement, design, and implementation, may not be clearly separated in projects using prototyping and incremental processes. We called back some respondents to ask whether they had specific software architecture in mind before the build vs. buy decision and COTS component selection. Their answers gave future clarification on whether COTS components were selected in the requirements or design phases.

The fourth possible threat to the internal validity is that some projects might not tailor the process because their processes already support COTS usage. So, we might miss some necessary process improvements that have been performed before the project start because we asked only salient process changes within the studied projects.

We selected ten projects in ten companies. However, we selected two projects in each of the other three companies. The fifth possible threat is that they are the same instead of different. However, data in Appendix A showed that all projects in the same company are different:

- In the first company, one project used waterfall and another used incremental process with prototyping. The project members were totally different, and the COTS selection and strategy to integrate COTS were also different.
- In the second company, one project used waterfall and another used incremental process with prototyping. The project members were totally different. The architectures of the final systems were also different.
- In the third company, the first project was finished two years before the second one started. The project members were totally different. The COTS selection method and strategy were also different.

6.3.3. External Validity

The primary threat to external validity is that the study is based on few and possibly not typical companies in Norway. According to Norwegian "Census Bureau" (SSB) (SSB, 2004), the mean value of employee number in Norwegian IT companies is six. The first possible external validity to this study is that our selected companies are mostly medium and large sized. However, the selected projects covered several application domains.

The second possible threat to external validity is that the size (based on person-hours) of the projects might bias the conclusion of this study. We have divided the projects into three groups: small (less than 10,000 person-hours), medium (between 10,000 and 100,000 person-hours), and large (more than 100,000 person-hours). We found out that the actual process used, the problems encountered, and the good practice crossed small, medium, and large projects.

The third possible threat to external validity is that most COTS components are well-defined and provided only secondary (support) functionality of the delivered system. The process customizations may be different if the COTS components are intended as the core part of the system, or the COTS components are large packages, such as ERP and content management systems.

7. Conclusions and Future Work

This paper has presented an exploratory study of COTS-based development processes in 16 software projects in 13 Norwegian IT companies. We conclude that using COTS components can be done *as part of traditional development processes* (e.g., waterfall and evolutionary) – there is no special "COTS-based development process." However, successful use of COTS components in such processes requires that some *new activities and roles* are introduced in order to reduce risks. Typical new activities are the build vs. buy decision, COTS component selection, and COTS component integration. A new role is that of a knowledge keeper. Two of the new activities, *the build vs. buy decision* and the *COTS component selection*, can be placed in different development phases (requirements, design, or implementation). This depends on project context, especially on the tacit familiarity with possible COTS components and the flexibility of requirements.

Although there were several different COTS component selection processes proposed from academia, we discovered that two COTS component selection processes were popular in practice. The first one is *familiarity-based* selection process. If the project members or external experts had enough previous experience with the candidate COTS component, their preference will decide which COTS component will be used in the new project. The second selection process is a *process combining Internet searches with hands-on trials.* Although several formal and complex COTS selection processes have been proposed, they were not used in this study. To reduce time-to-market, industrial practitioners performed only the most necessary steps in COTS component selection and made the final decision based on the results of the hands-on trial.

A set of explanatory scenarios, encountered problems, and examples of good practices have been synthesized to assist in the customization of the traditional

development processes with the new activities and roles. Although some of these problems and examples of good practice have been mentioned in other previous studies, we discovered several new ones, which decided the success or failure of the studied projects and deserve to be investigated further.

The main limitation of this research is that it is based only on software development projects in Norway, and that the sample size is small and perhaps not representative in profile for a typical COTS-based project. In the future, we will investigate further *the actual COTS-based development processes*. Although we proposed some possible customization of COTS-based development processes based on the results of this study, more samples are needed to validate them. The validation will focus on when, how, and why the new activities (build vs. buy decision, and COTS component selection) are performed.

We will also do a further study on *systematic risk management in COTS-based development.* The different process scenarios may meet different risks and need different risk mitigation methods. To manage risks in COTS-based development, it is necessary to identify risks, evaluate them and design risk mitigation strategies. Although several risk management methods have been proposed by case studies, there are few studies to verify these proposals and investigate how to use them in different process scenarios.

We are now formulating more explicit research hypotheses based on the findings of this study and the lessons learned. We plan to design a web-based questionnaire and use this to perform a quantitative survey in a large sample in Norwegian IT companies, in parallel with replicating the study in Germany and Italy.

Acknowledgments This study is supported by the INCO, SPIKE, and FAMILIES projects. Comments from our colleague Tore Dybå gave valuable input to this study. We thank the colleagues in these projects, and all the participants in this study.

Appendix A. Detailed Information on Process Customization

Comp. ID	Proj. ID	Succ. or failed	Main process	Other activities added	New role added
C1	P1	Failed	Waterfall	Buy vs. build. Selection. Build glueware & addware.	External expert: Hired one person from the vendor as consultant.
C1	P2	Failed	Incremental & prototyping	Customer decided COTS. Learning. Build glueware.	
C2	P3	Succ.	Incremental & prototyping	Buy vs. build. Selection. Learning. Build glueware & addware.	Internal expert.
C2	P4	Succ.	Waterfall	Buy vs. build. Selection. Learning. Build glueware & addware.	Internal expert.
C3	P5	Succ.	Incremental & prototyping	Buy vs. build. Selection. Learning. Build glueware.	
C3	P6	Succ.	Incremental & prototyping	Buy vs. build. No selection as it was the only alternative. Learning. Build glueware.	

Appendix A (continued)

Comp. ID	Proj. ID	Succ. or failed	Main process	Other activities added	New role added
C4	P7	Succ.	Incremental & prototyping	Buy vs. build. Selection. Learning. Build glueware & addware.	
C5	P8	Succ.	Incremental & prototyping	Buy vs. build. Selection. Build addware.	Internal expert.
C6	P9	Succ.	Incremental & prototyping	Without build vs. buy decision because using COTS is a company strategy. Selection. Learning. Build glueware & addware.	Internal expert.
C7	P10	Failed	Waterfall	Buy vs. build. Selection. Build glueware Learning.	External expert: The integration was outsourced to a consulting company.
C8	P11	Succ.	Incremental & prototyping	Buy vs. build. Selection. Learning. Build glueware.	Internal expert to follow COTS evolution.
C9	P12	Succ.	Waterfall & prototyping	Buy vs. build. Selection. Learning. Build glueware.	
C10	P13	Succ.	Waterfall	Buy vs. build Selection. Learning. Build glueware.	Internal expert.
C11	P14	Failed	Waterfall	Buy vs. build. Selection. Learning. Build glueware.	
C12	P15	Succ.	Incremental & prototyping	Buy vs. build. Selection. Learning. Build glueware.	Internal expert.
C13	P16	Succ.	Incremental & prototyping	Buy vs. build. Selection. Learning. Build glueware.	Internal expert.

Appendix B. Detailed Information on COTS Component Selection

Proj. ID	COTS ID	Selection process	In which phase was COTS selected
P1	1	Demo (recommended from external expert).	Implementation
P1	2	Demo (recommended from external expert).	Implementation
P2	3	Decided by customer	Design
P3	4	Experience	Design
P3	5	Internet search with pilot	Requirements
P3	6	Experience	Design
P4	7	Internet search with pilot	Requirements
P4	8	Experience	Requirements
P4	9	Experience	Requirements
P5	10	Experience	Implementation
P5	11	Internet search with pilot	Implementation

Appendix B (continued)

Proj. ID	COTS ID	Selection process	In which phase was COTS selected
P6	12	Experience	Requirements
P7	13	Internet search with pilot	Design
P8	14	Experience	Implementation
P8	15	Experience	Implementation
P9	16	Experience	Design
P9	17	Experience	Design
P9	18	Experience	Design
P10	19	Demo.	Design
P10	20	Demo.	Design
P10	21	Demo.	Design
P11	22	Internet search with pilot	Design
P11	23	Internet search with pilot	Design
P12	24	Experience	Design
P13	25	Experience	Requirements
P14	26	Demo.	Implementation
P15	27	Experience	Design
P15	28	Recommended from external expert	Requirements
P16	29	Experience	Design
P16	30	Experience	Design

References

Abts C, Boehm BW, Clark EB (2000) COCOTS: A COTS software integration cost model—model overview and preliminary data findings. Proc. of the 11th ESCOM Conf, Munich, Germany, pp 325–333

Albert C, Brownsword L (2002) Evolutionary process for integrating COTS-based system (EPIC): an overview. SEI, Pittsburgh, Available at: http://www.sei.cmu.edu/publications/documents/02.reports/02tr009.html

Alves C, Fineklstein A (2003) Investigating conflicts in COTS decision-making. JSEKE 13(5):473–493

Basili VR, Boehm BW (2001) COTS-based system top 10 lists. IEEE Computer 34(5):91–93

Boehm B (2000) Requirements that handle IKIWISI, COTS, and rapid change. IEEE Computer 33(7):99–102

Boehm BW, Abts C (1999) COTS integration: plug and pray? IEEE Computer 32(1):135–138

Boehm BW, Port D, Yang Y, Bhuta J (2003) Not all CBS are created equally COTS-intensive project types. Proc. of the 2nd Int Conf on COTS-Based Software Systems (ICCBSS'03), Ottawa, Canada, Springer, LNCS Vol. 2580, pp 36–50

Brownsword L, Oberndorf T, Sledge C (2000) Developing new processes for COTS-based systems. IEEE Software 17(4):48–55

Carney D, Long F (2001) What do you mean by COTS? Finally, a useful answer. IEEE Software 7(2):83–86

Chung L, Cooper K (2002) A knowledge-based COTS-aware requirements engineering approach. Proc. of the 14th Int Conf on Software Engineering and Knowledge Engineering, Ischia, Italy, pp 175–182

ComponentSource (2004) Available at: http://www.componentsource.com/

COTS Lessons Learned Repository (2004) Available at: http://fc-md.umd.edu/ll/index.asp

Crnkovic I, Hnich B, Jonsson T, Kiziltan Z (2002) Specification, implementation, and deployment of components. Communication of the ACM 45(19):35–40

FAA Guide (2003) Federal Aviation Administration, Software Engineering Resource Center. FAA COTS risk mitigation guide: practical methods for effective COTS acquisition and life cycle support. Available at: http://www.faa.gov/aua/resources/cots/Guide/CRMG.htm

FAMILIES Project (2003) Available at: http://www.esi.es/en/Projects/Families
Fowler FJ Jr. (2001) Survey research methods (Applied social research methods). 3rd edition, Thousand Oaks, CA, USA: Sage
Fox G, Lantner K, Marcom S (1997) A software development process for COTS-based information system infrastructure. Proc. of the 5th Int Symposium on Assessment of Software Tools (SAST '97), Pittsburgh, Pennsylvania, USA, pp 133–142
INCO Project (2000) Available at: http://www.ifi.uio.no/~isu/INCO/
Kontio J (1996) A case study in applying a systematic method for COTS selection. Proc. of the 18th Int Conf on Software Engineering, Berlin, Germany, pp 201–209
Kotonya G, Rashid A (2001) A strategy for managing risk in component-based software development. Proc. of the 27th EUROMICRO Conference, Warsaw, Poland, pp 12–21
Lawlis PK, Mark KE, Thomas DA, Courtheyn T (2001) A formal process for evaluating COTS software products. IEEE Computer 34(5):58–63
Leung KRPH, Leung HKN (2002) On the efficiency of domain-based COTS product selection method. Journal of Information and Software Technology 44(12):703–715
MacCrimmon KR (1973) An overview of multiple objective decision making. Proc. of the Multiple Criteria Decision Making, University of South Carolina, pp 18–44
Maiden NAM, Ncube C (1998) Acquiring COTS software selection requirement. IEEE Software 15(2):46–56
Maiden NAM., Kim H, Ncube C (2002) Rethinking process guidance for selecting software components. Proc. of the 1st Int Conf on COTS-Based Software Systems (ICCBSS'02), Orlando, Florida, USA, Springer, LNCS VOL. 2255, pp 151–164
Morisio M, Tsoukias A (1997) IusWare: a methodology for the evaluation and selection of software products. IEE Proceedings—Software Engineering 144(3):162–174
Morisio M, Seaman CB, Parra AT, Basili VR, Kraft SE, Condon SE (2000) Investigating and improving a COTS-based software development process. Proc. of the 22nd Int Conf on Software Engineering, Limerick, Ireland, pp 31–40
Morris E, Albert C, Brownsword L (2003) COTS-based development: taking the pulse of a project. Proc. of the 2nd Int Conf on COTS-Based Software Systems (ICCBSS 2003), Ottawa, Canada, Springer LNCS, VOL. 2580, pp 168–177
NAICS (2004) Available at: http://www.revenue.state.ne.us/tax/current/buscodes.pdf
Ncube C, Dean JC (2002) The limitation of current decision-making techniques in the procurement of COTS software components. Proc. of the 1st Int conf on COTS-Based Software Systems (ICCBSS'02), Orlando, Florida, USA, Springer, LNCS vol. 2255, pp 176–187
Ochs M, Pfahl D, Diening GC, Kolb BN (2001) A method for efficient measurement-based COTS assessment and selection—method description and evaluation results. Proc. of the 7th Int. Software Metrics Symposium, London, England, pp 285–297
Robson C (2002) Real world research: a resource for social scientists and practitioner-researchers (Regional Surveys of the World), 2nd edition. Oxford, Blackwell
Rose LC (2003) Risk management of COTS based systems development. In: Component-Based Software Quality, Springer LNCS, vol. 2693, pp 352–373
Saaty TL (1990) How to make a decision: the analytic hierarchy process (AHP). Eur J Oper Res 48(1):9–26
SPIKE project (2002) Available at: http://www.idi.ntnu.no/grupper/su/spike.html
SEI (Software Engineering Institute) (2004) COTS-based initiative description. Available at http://www.sei.cmu.edu/cbs/cbs_description.html
SSB: Norwegian Census Bureau (2004) Available at: http://www.ssb.no
Strauss A, Corbin JM (1998) Basics of qualitative research: grounded theory procedures and techniques. 2nd edition. Thousand Oaks, CA, USA: Sage
Torchiano M, Morisio M (2004) Overlooked aspects on COTS-based development. IEEE Software 21(2):88–93
Tran V, Liu DB, Hummel B (1997) Component-based systems development: challenges and lessons learned. Proc. of the 8th IEEE Int. Workshop on Software Technology and Engineering Practice, London, UK, pp 452–462
Voas J (1998a) COTS software—the economical choice? IEEE Software 15(2):16–19
Voas J (1998b) The challenges of using COTS software in component-based development. IEEE Computer 31(6):44–45

Jingyue Li is presently a Ph.D. student in computer science at the Norwegian University of Science and Technology (NTNU). He received his master's degree in computer science from Beijing Polytechnic University in 2001. He worked at IBM China Ltd., before he joined NTNU. His research interests include software engineering, commercial-off-the-shelf based development, open source based development, project risk management, and aspect-oriented programming.

Finn Olav Bjørnson received his master's degree in computer science from the Norwegian University of Science and Technology (NTNU) in Trondheim in 2002. He is currently working on his PhD degree in Software Engineering at the Department of Computer and Information Science at NTNU. The focus of his research is on the use of knowledge management theories to explain and understand software process improvement efforts.

Reidar Conradi received his MS and his PhD degrees from the Norwegian University of Science and Technology in Trondheim (NTNU, previously called NTH), and has been at NTNU since 1975. He is now a professor at the Department of Computer and Information Science at NTNU. Conradi was a visiting scientist at Fraunhofer Center for Experimental Software Engineering in Maryland and at Politecnico di Milano in 1999 to 2000. His research interests include software quality, software process improvement, software engineering databases, versioning, component-based development (OSS and COTS) and reuse, mobile and distributed systems and empirical studies related to the mentioned topics.

Vigdis B. Kampenes received her MSc degree in statistics from the University of Trondheim in 1993 and is now working on a PhD degree in software engineering at the University of Oslo and Simula Research Laboratory. She has eight years experience as a statistician in the pharmaceutical industry and two years as a consultant in the IT industry. Her main research interest is research methods in empirical software engineering.