An Empirical Study of Variations in COTS-based Software Development Processes in Norwegian IT Industry

Jingyue Li [1], Finn Olav Bjørnson [1], Reidar Conradi [1,2], Vigdis By Kampenes [2]

[1] *Dept. of Computer and Information Science*
Norwegian Univ. of Science and Technology
NO-7491 Trondheim, Norway
{jingyue,bjornson,conradi}@idi.ntnu.no

[2] *Simula Research Laboratory*
P.O.BOX 134, NO-1325 Lysaker, Norway
{vigdis}@simula.no

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 to investigate how to use and customize them to different project contexts. This paper describes an exploratory study of state-of-the-practice of COTS-based development processes. 16 software projects in 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 and lessons learned, and can be used to customize the actual development processes, e.g. in which lifecycle phase to put the new activities. Such customization crucially depends on project context, such as previous familiarity with possible COTS components and flexibility of requirements.

1. Introduction

COTS-based development has become more and more important in software and system development. COTS usage promises faster time-to-market and increased productivity [17]. At the same time, COTS-based software introduces many risks: unknown quality properties of the chosen COTS components can be harmful for the final product, or economic instability of the COTS vendor may terminate the maintenance support of its COTS components [18].

The use of COTS components introduces new system circumstances, which again 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 [12, 14], or similar large projects. 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, and in small or medium-sized projects.

This study has investigated the commonalities and differences between development processes in 16 COTS-based software projects in Norway. It also summarized process scenarios in successful vs. unsuccessful COTS-based projects. 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. Results are presented in Section 4 and discussed in Section 5. Conclusion and future research are presented in Section 6.

2. Related work

Typically, a COTS-based process consists of four phases, comprising [1]:

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

There is a consensus that the use of COTS component implies changes in the software process [20]. Most studies on COTS-based development process focus on two dimensions:

- Process of the whole software development lifecycle.
- Process of the specific phase, especially in COTS component selection and evaluation.

Proceedings of the 10th International Symposium on Software Metrics (METRICS'04)
1530-1435/04 $ 20.00 IEEE

2.1. Process of the whole software development lifecycle

Boehm et al. [3] regard both the waterfall model and evolutionary development as unsuitable for COTS-based development because:

- In the waterfall model, requirements are identified at an earlier stage and the COTS components 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 lack of code availability bars the development team from adjusting them to their needs.

Based on the above arguments, they proposed that development models, which explicitly take risk into account, are more suitable for COTS-based development than the traditional waterfall model 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) [13]. The proposed process integrates COTS component 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 system with COTS components for many years. Their experiences have been captured by Morisio et al. in [12]. This team of people investigated the various processes that were used across 15 projects at NASA and then they developed a COTS-based development process that was the most representative of the processes used by the projects.

Torchiano and Morisio [16] have interviewed seven small and medium sized software companies in Norway and Italy. Their results are summarized in six theses, and challenge many "assumed" truths about COTS-based development:

- T1: Open source software is often used as closed source.
- T2: Integration problems result from lack of compliance with standards; architectural mismatches constitute a secondary issue.
- T3: Custom code mainly provides additional functionalities (i.e. addware, not glueware).
- T4: Developers seldom use formal selection procedures. Familiarity with either the product or the generic architecture is the leading factor in selection.
- T5: Architecture is more important than requirements for product selection.
- T6: Integrators tend to influence the vendor on product evolution whenever possible.

2.2. COTS component selection and evaluation process

Based on case studies, researchers have proposed several COTS component selection processes.

One kind of process is based on formal decision making algorithms [5]. It includes three basic elements: selecting evaluation criteria (factors), collecting and assigning values to these criteria, and applying formal decision making algorithms such as MAUT [21] or MCDA [22]. Maiden et al. have proposed a meta-model selection process [11]. It is similar to the previous decision making processes, but invokes a software tool to infer properties about the current status of the selection process and to recommend relevant process guidance.

RCPEP [10] is a requirement-driven COTS component evaluation process. It identifies every product that possibly addresses the requirements, conducts a trade study to narrow the list of serious candidates, and evaluates the remaining candidates using hands-on scenarios.

3. Research approach

Most studies on COTS-based development processes are limited to propose new technology (e.g. revised processes) without empirical evidence, or to discover existing practice (e.g. component selection) from case studies. Empirical studies are few and mostly based on a small project sample or on projects with rather similar contexts, e.g. 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. Such a context involves developer skills, software technologies, application domain, and requirements "toughness" etc.

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 to 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

mainly on the development phases, not on software maintenance and evolution.

3.1. COTS component definition

The essential question for COTS–based development is "What do you mean by a COTS component?" There are many different definitions of COTS components [2, 4].We have used the definition by Torchiano and Morisio [16], where a COTS component:

- Is either provided by some other organizations in the same company, or provided by external companies as a commercial product.
- Is integrated into the final delivered system.
- Is not a commodity, i.e. not shipped with an operating system, not provided with the development environment, not generally included in any pre-existing platforms.
- Is not controllable by the user, in terms of provided features and their evolution. Our addition: This normally means "black box", i.e. no source code available.

The granularity of a COTS component can be different. Some regard 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 [7]. That is, a component is an "*Executable unit of independent production, acquisition, and deployment that can be composed into a functioning system.*" This definition means that we include both 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 [6].

3.2. Research questions

3.2.1. Research question RQ1. Boehm et al. [3] 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:

RQ1: What was the actual process used in the projects using COTS components?

3.2.2. Research question RQ2. Furthermore, Morisio et al. [12] 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. It is important to investigate under what conditions each variation is most suitable [12]. However, there have been no further studies on this so far. Our second research question is then:

RQ2: What are the commonalities and possible variations in COTS-based development processes?

3.2.3. Research question RQ3. Finally, all previous studies on COTS-based development processes have emphasized the risks issues [3, 12, 13]. To successfully customize COTS-based processes, it is important to summarize and understand the experiences of 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:

RQ3. What are process scenarios (variations) of the projects using COTS component successfully and not successfully? What are the relationship between possible risks and process scenarios?

3.3. Overall research method

Our initial plan was to make a quantitative survey using pre-stated hypotheses with standardized questions in a formal questionnaire. This requires substantial work to give consistent and reliable results. We instead decided to go for an exploratory pre-study after two rounds of pre-tests on the questionnaire. This employed more open research questions, not fixed hypotheses, and with a rather detailed interview guide.

Our main investigation tool in this study was *semi-structured interviews*. We used a modified version of our drafted questionnaire to serve as an interview guide. The interview guide included a mixture of specific and open-ended questions. The specific ones were used to solicit information on project context. For example, one question used the application domains of the final system (product) based on the North American Industrial Classification System (NAICS) [19]. The open-ended questions were used to gather information on actual COTS-based development processes. The questions covered the main (existing) process being used, 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 COTS component. We then asked when and how did they do these changes if they added or changed some activities.

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

- 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 increase the construct validity of the interviews, we did two rounds of pre-test in ten other projects. These projects were selected based on convenience. For this, we made an introductory letter to first be 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.

In the final version of the interview guide, we added three meta-questions following each question. These meta-questions asked the respondents, if the question was clear, if they needed clarification, and if they could give specific comments if the question was confusing. All this is standard procedure from social science [24].

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. During this pre-process, 16 companies (among 34 such – see below) found out that their projects did not match our definition on COTS-based project, and therefore could not join the study.

3.4. Data collection

Interviews and data collection were conducted by two PhD students, Li and Bjørnson, from NTNU from Nov. 2003 to Jan. 2004. They contacted 34 IT companies, where most were involved in two Norwegian R&D Projects (INCO [9], and SPIKE [15]) and one EU R&D project (FAMILIES [8]). 16 of them could not join the survey because they had no proper COTS-based projects. 13 of these 16 companies used only in-house build components, and 3 of them used only open source components. There was one other company that could not participate in the research because of confidential issues. For the rest of the 17 companies, 13 of these volunteered to join in the study (as mentioned in 3.3 above). The 16 projects were selected from these 13 companies. The total sample was based on convenience, but we ensured that we had included respondents from large, medium and small companies.

The corresponding interview was aimed to take 1-1.5 hour, i.e. longer than filling-out a similar questionnaire. 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 confidential.

As in the pre-tests, respondents were first contacted by phone and email, reusing the introductory letter from the pre-tests. If they had relevant COTS-based projects, and would like to assign effort to the survey, the interview guide was sent to them a few days before the personal interview. In this way, the respondents were well-prepared and could select a proper project they had been involved in. 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 Ph.D. students (none with two interviewers), took from 60 to 80 minutes, and was taped. However, 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, could take weeks.

3.5. 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 RQ1** is to investigate the actual development processes used in the project. The data was analyzed by constant comparison method [23]. As all respondents had clear definition on the main process used in their project, these processes were early grouped into traditional process (i.e. waterfall, prototyping, or incremental) or totally new process.

- The purpose of the second **research question RQ2** is to see the similarities and differences between process changes in the studied projects. The main analysis method used is cross-case analysis [23], 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 proposition 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. And then 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 evidences (projects).
- The purpose of the third **research question RQ3** is to summarize the process scenarios of the successful and unsuccessful projects. We first group projects into two groups based on the COTS component's positive or negative effect to 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 was compiled of short phases that describe not only the process changes in each project, i.e. new or changed activities, when and how these changes were performed. We also recorded project contexts, i.e. the developers' experience with COTS components. In all this, we also recorded the corresponding lesson learned.

4. Survey results

4.1. Companies

As mentioned, we interviewed 16 projects in 13 companies. All the companies are Norwegian IT companies. 9 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. 6 companies are IT consultancies, 5 are software vendors, and 2 are telecom companies. 5 of these companies are publicly traded companies, the others are privately held. Some company background information is listed in the following Table 1.

Table 1. Background information of the 13 companies

Staff Size	Type	Ownership	Main business area
5	Stand-alone	Privately held	Software Vendor
17	Stand-alone	Privately held	IT Consulting
36	Stand-alone	Privately held	IT Consulting
42	Stand-alone	Privately held	IT Consulting
50	Subsidiary	Publicly traded	Software Vendor
60	Subsidiary	Privately traded	IT Consulting
80	Stand-alone	Privately traded	Software Vendor
100	Stand-alone	Privately held	Software Vendor
120	Stand-alone	Publicly traded	Software Vendor
130	Stand-alone	Privately held	IT Consulting
275	Subsidiary	Publicly traded	Telecom Industry
320	Subsidiary	Publicly traded	Telecom Industry
500	Stand-alone	Publicly traded	IT Consulting

4.2. Respondents

We had one respondent from each project. 3 respondents are IT managers, 4 are project managers, 5 are software architects, 3 are software developers, and 1 is a software development researcher. 9 of them have more than 10 years of software development experience, and 12 of them have more than 4 years working experience with COTS-based development. 8 of them have a master degree, and the rest have a bachelor degree. 12 of them have a principal degree on 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 one company. We also 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 of the 16 projects.

Table 2. Background information of the 16 projects

Effort (person-hours)	Main development language	Application domain
700	C, Java	Scientific
2,400	Java	Agriculture
3,000	Visual Basic	El. Power
7,000	Java	Finance
8,000	C++	El. Power
8,000	C++	Finance
10,000	C++	Construction
15,000	C++	Scientific
16,000	C++	Internet Service provider
23,000	C++	Telecom.
30,000	Power Builder	Finance
42,000	C	Scientific
58,000	C++, Java	Telecom
63,000	C++, VB	Whole trade
92,000	Ada, Java	Transport
128,000	C++	Telecom

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 maximum three typical COTS component 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 the 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 the final system to the customer.

Table 3. Background information of some COTS component

COTS component name	Functionality
Staffware	Workflow management
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

4.5. Results of research questions

4.5.1. Research question RQ1. RQ1 is to investigate what were the actual processes used in projects with COTS components. To study this issue, we asked two open questions: we first ask the respondents to describe the development processes they used. We also asked if they selected the main development processes because of using COTS components.

Results of the first question show that the main development processes of the surveyed projects can be grouped into three categories: *pure waterfall, waterfall mixed with prototyping, incremental mixed with prototyping*. Five surveyed 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 intent 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 use of COTS component.

Therefore, the answer to research question RQ1 is: the so-called COTS-based development process is to customize the traditional development process due to the use of COTS components.

4.5.2. Research question RQ2. RQ2 is to investigate 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 because of using COTS components. Results show that 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 surveyed 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 processes and criteria in their COTS component selection.
- ***Learn and understand COTS component*:** Another new activity is to learn and understand COTS component. 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 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 projected built more or less glueware (the code to integrate the COTS component) and/or addware (the code to complete the required functionality of COTS component) to integrate COTS component with the system and with other components in the system.

The possible added new role is:

- ***COTS component knowledge keeper*:** As mentioned above, it was difficult to really understand the COTS component. In 12 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 COTS components 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. 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. proposed a two-phase build vs. buy decision. The first decision is in the requirement phase and the second in the design phase [12]. Our results indicate that only 2 of the 5 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 quite familiar with the possible COTS components.
- ***The COTS component selection and evaluation process:*** The selection process in our studied projects can be summarized into two categories:

1) SP_fam: Familiarity-based selection process: The previous familiarity came either from the project members inside projects, i.e. the COTS component knowledge keeper, or from a colleague outside the project. If the project members had enough previous experience on the candidate COTS components, this experience will be the key factor in the COTS component selection. 16 of the 30 given COTS components were selected based on the suggestion of the COTS component knowledge keeper inside the projects. 4 COTS components were selected based on recommendations from colleagues outside the project or organization.

2) SP_unfam: Internet search, trial-based selection process: 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 etc.

Step 3: After a small number of candidate COTS components had been decided, they

downloaded a demo version of these from the web and test the COTS components.

This kind of selection is similar to the process proposed by Lawlis et al. [10]. The main difference is that the actual process was much faster. Developer 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 bulletin had a good chance to be selected, because they were assumed to be better tested and in general of high quality.

The answer to research question RQ2 is: there are some common new activities and one new role added in COTS-based development process. The possible variations are when and how to perform these new activities.

4.5.3. Results of RQ3. RQ3 is about summarizing the process scenarios of the successful and unsuccessful COTS-based projects. The focus is to investigate the relationship between the possible risks and the development processes. We asked respondents to summary their lessons learned and experiences in the studied projects. Their answers cover several viewpoints such as business, technology, and process. We extracted only process-relevant parts, i.e. what were the main development processes, and when and how were the new process activities performed.

In our definition, a successful COTS-based project means that the COTS components contributed positively to time-to-market and system quality. Otherwise, it is regarded as unsuccessful.

Four of the 16 studied projects were regarded as unsuccessful. The scenarios and lessons learned are summarized into scenario Sc1 to Sc3 in Table 4.

Table 4. The scenarios of the 4 unsuccessful COTS-based projects

ID	Characterizations	Lesson learned
Sc1	Two projects used waterfall processes. Made the build vs. buy decision and the COTS component selection in the implementation phase. Selected unfamiliar COTS components.	**Lesson 1**: The COTS component selection should be implemented no later than the design phase in the waterfall processes. **Lesson 2**: They should inform the customer after they decided to use COTS components.
Sc2	One project used waterfall process. Selected COTS components in design phase but mainly on their functionalities. Selected unfamiliar COTS components.	**Lesson 3**: They should consider more on how to integrate the COTS components. If the COTS components were hard to be integrated, they should build alternative components themselves.
Sc3	One project used incremental & prototyping process. Selected the COTS components in design phase but mainly on their functionalities. Selected unfamiliar COTS components.	**Lesson 3**: See above

The backgrounds of each lessons learned are following:

- ***Lesson 1:*** The two waterfall projects 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.
- ***Lesson 2:*** Although some previous studies proposed to (re)negotiate requirements with the customer when developers found later limitations of COTS components, none of the two waterfall two projects (re)negotiated the requirements with customers even they wanted to. This is because the developer wanted to use COTS components, not the customer. The customer cares only of the final result. If developers were not allowed to use COTS components, (re)negotiation of requirements due to COTS limitations is difficult.
- ***Lesson 3***: The project members did not knew 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 COTS component selection and recommended a certain COTS component that could provide the needed functionality. This COTS component caused a lot of quality problems.

12 projects successfully integrated the COTS components. The scenarios can be summarized into scenario Sc4 to Sc6 in the following Table 5.

Table 5. The scenarios of 12 successful COTS-based projects

ID	Characterization	Lessons learned
Sc4	Two projects used waterfall process. Made the build vs. buy decision and selected COTS components in the requirement phase. Selected familiar COTS components.	**Lesson 4**: Selected the COTS components mainly based on the easiness of integration.
Sc5	One project used waterfall with some prototyping process. Made the build vs. buy decision and selected COTS components in design phase. Selected familiar COTS components.	**Lesson 4**: See above
Sc6	Nine projects used incremental with prototyping processes. Seven projects selected COTS components in requirement with familiar COTS components. Two projects selected COTS component in design phase with unfamiliar COTS components.	**Lesson 5:** Using COTS components in the incremental & prototyping process helped the success of the project. **Lesson 6:** Rank the integration of unfamiliar COTS components as high risk task and integrate such components before other components.

The detailed explanations of each lessons learned are following:

- ***Lesson 4****:* 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.
- ***Lesson 5***: Seven projects started to select COTS components in the requirement phase. This makes it simpler to match and possibly negotiate customers' (volatile) requirements with available COTS components functionality, because the customer will accept the COTS components functionality showed in the prototype. In the incremental with prototyping process, the main benefit of using COTS components is to have a fast building of a prototype. They could also learn and understand how to use the COTS components in building such a prototype.
- ***Lesson 6****:* In incremental with prototyping process, integrating unfamiliar COTS components was ranked as a high risk task. These unfamiliar components had been integrated and tested before other components.

The answer to research question RQ3 is: COTS-based development could be performed successfully using waterfall and evolutionary processes. However, different process scenarios may face different risks.

5. Discussion

5.1. Comparison with related work

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

- Boehm et al. [3] regard both the waterfall model and evolutionary development as unsuitable for COTS-based development. Results of research question RQ3 showed however that some projects could integrate COTS components successfully, using waterfall or evolutionary processes. However, the results of RQ3 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 et al. In addition, results of RQ1 showed that the main development processes were decided *before* the decision of whether to use COTS components or not. So, if the process is risk-driven as Boehm et al. proposed [3], developers should use non-COTS related risks to decide the main development processes first, and start to consider COTS related risks after they decide to use COTS components.
- Both EPIC [13] and process proposed by Morisio et al. [12] 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. Results of RQ2 gave farther support to

their conclusions. This study shows that there are several possible variations in COTS-based development process.

- Although there are several possible processes in COTS component selection [5, 10, 11], results of RQ2 showed that none of the studied projects used formal decision-making algorithms [5, 11]. If developers had enough previous experiences with specific components, COTS component selection was based on their previous experience (see **SP_*fam*** in above section 4.5.2). If they had no previous experience on the selected COTS components, the selection was based on hands-on trials (see **SP_*unfam*** in above section 4.5.2). The selection process was similar with requirement-driven process [5].

5.2. Customizing the COTS-based development process

Our results show that the COTS-based development process is not a totally new and standardized process. It is a customization (change or add new activities) on actual traditional development process and project members' familiarity with relevant COTS components. Based on 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 only on them. The process design could include two elements:

- First, decide the main development processes based on project contexts. The main development process could be decided based on some risk considerations as proposed by Boehm et al. [3]
- If the main development process is waterfall or evolutionary, the processes could be customized based on the actual main development process and project members' familiarity with relevant COTS components as following Figure 1. For each customization, some possible risks must be identified and managed as we discovered from this study.

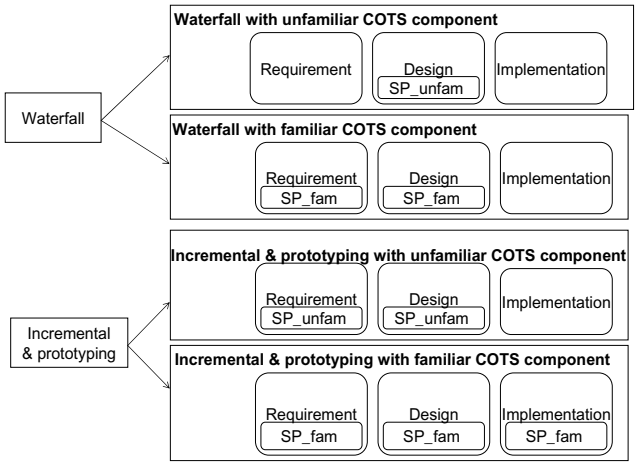

Figure 1. Possible COTS-based development process customizations

SP_fam: Familiarity-based selection process
SP_unfam: Internet search, trial-based selection process

Waterfall with unfamiliar COTS components*:* The scenario Sc1 and Sc2 showed that it is difficult to use an unfamiliar COTS component successfully if the general process is waterfall. The risk is that it is difficult to find COTS components to fully satisfy the requirements after the requirement and design have been decided upon. So, 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 SP_unfam, and the ease of integration should be given much consideration in the make vs. buy decision. If the candidate COTS components do not appear to be easy integrate, it is better to develop them in-house. Another recommendation is to inform the customer; if possible, of 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.

Waterfall with familiar COTS components*:* If the project members are quite familiar with relevant COTS components, the make 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.

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 integration first, if there are different phases to implement the system incrementally.

Incremental & prototyping with familiar COTS components: If the project members have enough experience on relevant COTS components; it is easy to build a prototype. In this case, our result showed that buying COTS components definitely had more advantages than to build them in-house. Here, the make vs. buy decision can be performed in the requirements and/or design phase. If there are different phases to implement the system incrementally, they don't need to implement the COTS components relevant composition first. The process could be the same as a non-COTS one.

5.3. Threats to validity

5.3.1. Internal validity. The responses to the three meta-questions show that the questions in the interview guide were understood consistently. In general, we think that the respondents have answered truthfully and comprehensively. Listening to the tape also helped ensure correct interpretation of answers and comments. However, having an independent (third) person to listen to the tape might have increased data quality.

5.3.2. External validity. The primary threat to external validity is that the survey is based on few and possibly not typical companies in Norway. However, some companies were subsidiaries of large world-wide companies. The projects selected also covered several application domains. Another threat is that most COTS components provided only support functionality of the system (product). The process customizations maybe different if the COTS components are intended as the core part of the system.

5.3.3. Construct validity. In this study, most variables are taken directly, or with little modification, from existing literature. We also did two rounds of pre-test in ten local IT companies. 10% of the questions and alternatives in the final interview guide were revised based on such comments. One possible threat to the construct validity is that the lifecycle phases, such as requirement, design, and implement, may not be clearly separated in projects using prototyping and incremental processes. We called back some respondents to ask whether they had a 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. Another possible threat to construct validity is our chosen success criteria for a COTS-based project, 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.

5.3.4. Conclusion validity. This study is a qualitative exploratory study. Future studies will be implemented to confirm the results of this study. On the other hand, the exploratory nature of the study uncovered several undocumented and seemingly common process variations.

6. Conclusion and future work

This paper has presented an exploratory study of COTS-based development processes in 16 software projects in Norwegian IT companies. The main findings are:

- Use of COTS components can be done as *part of traditional development processes* (e.g. waterfall and evolutionary) – there is no special "COTS-based development process".
- Successful use of COTS components in such processes, however, 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 COTS component selection processes have been verified in practice, i.e. *familiarity-based* selection process and process *combining Internet searches with hands-on trials*.
- 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 familiarity with possible COTS components and the flexibility of requirements.
- A set of explanatory scenarios and guidelines have been synthesized to assist in the customization of the traditional development processes with the new activities and roles.

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. In the future, we will focus on two main research questions:

- *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 verify them. The verifications will focus on when and how, and why the new activities (build vs. buy decision and COTS component selection) are performed.

- *More systematic risk management.* The different process scenarios may meet different risks and need different risks mitigations methods. Although several risk management methods have been proposed in the literature, there are few studies to verify these proposals and to 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 of around 75 representative Norwegian IT companies of different sizes and profiles, in parallel with replicating the study in at least Germany and Italy.

7. Acknowledgements

This study is supported by the INCO, SPIKE, and FAMILIES projects. We thank the colleagues in these projects, and all the participants in this study.

8. References

[1] Chris Abts, Barry W. Boehm, and Elizabeth Bailey Clark: COCOTS, "A COTS Software Integration Cost Model - Model Overview and Preliminary Data Findings", Proc. 11th ESCOM conference, Munich, Germany, April 2000, pp. 325-333.

[2] Victor R. Basili and Barry W. Boehm, "COTS-Based System Top 10 Lists", IEEE Computer, 34(5):91-93, May 2001.

[3] Barry W. Boehm and Chris Abts, "COTS integration: Plug and Pray?" IEEE Computer, 32(1):135-138, January 1999.

[4] David Carney and Fred Long, "What do You Mean by COTS? Finally, a Useful Answer", IEEE Software, 17(2):83-86, March/April 2000.

[5] Cornelius Ncube and John C. Dean, "The Limitation of Current Decision-Making Techniques in the Procurement of COTS Software Components", Proc. First International Conference on COTS-Based Software Systems (ICCBSS'02), Orlando, FL, USA, February 4-6, 2002, Springer Verlag LNCS 2255, pp. 176-187.

[6] ComponentSource: http://www.componentsource.com/.

[7] Ivica Crnkovic, Brahim Hnich, Torsten Jonsson, and Zeynep Kiziltan, "Specification, Implementation, and Deployment of Components", Communication of the ACM, 45(19):35-40, October 2002.

[8] FAMILIES project description, 2003, http://www.esi.es/en/Projects/Families.

[9] INCO project description, 2000, http://www.ifi.uio.no/~isu/INCO/.

[10] Patricia K. Lawlis, Kathryn E. Mark, Deborah A. Thomas, and Terry Courtheyn, "A Formal Process for Evaluating COTS Software Products", IEEE Computer, 34(5):58-63, May 2001.

[11] N.A.M. Maiden, H. Kim, and Cornelius Ncube, "Rethinking Process Guidance for Selecting Software Components", Proc. First International Conference on COTS-Based Software Systems (ICCBSS'02), Orlando, FL, USA, February 4-6, 2002, Springer Verlag LNCS 2255, pp. 151-164.

[12] M. Morisio, C.B. Seaman, A. T. Parra, V. R. Basili, S. E. Kraft, and S. E. Condon, "Investigating and Improving a COTS-Based Software Development Process", Proc. 22nd International Conference on Software Engineering, Limerick, Ireland, IEEE CS Press, June 2000, pp. 31-40.

[13] Cecilia Albert and Lisa Brownsword, "Evolutionary Process for Integrating COTS-Based System (EPIC): An Overview", SEI, Pittsburgh, 2002. Available at: http://www.sei.cmu.edu/publications/documents/02.reports/02tr009.html.

[14] Software Engineering Institute, "COTS-Based Initiative Description", SEI, Pittsburgh, 2004, available at http://www.sei.cmu.edu/cbs/cbs_description.html.

[15] SPIKE project description, 2002, http://www.idi.ntnu.no/grupper/su/spike.html.

[16] M. Torchiano and M. Morisio, "Overlooked Facts on COTS-based Development", IEEE Software, 21(2):88-93, March/April 2004.

[17] J. Voas, "COTS Software – the Economical Choice?", IEEE Software, 15(2):16-19, March/April 1998.

[18] J. Voas, "The challenges of Using COTS Software in Component-Based Development", IEEE Computer, 31(6):44-45, June 1998.

[19] NAICS classification of domains. Available at:http://www.revenue.state.ne.us/tax/current/buscodes.pdf.

[20] L. Brownsword, T. Oberndorf, and C. Sledge, "Developing New Processes for COTS-Based Systems", IEEE Software, 17(4):48-55, July/August 2000.

[21] MacCrimmon K. R., "An Overview of Multiple Objective Decision Making", Proc. Multiple Criteria Decision Making, University of South Carolina Press, 1973, pp. 18-44.

[22] M. Morisio and A. Tsoukias, "IusWare: a methodology for the evaluation and selection of software products", IEE Proceedings – Software Engineering, 144(3): 162-174, June 1997.

[23] Carolyn B. Seaman, "Qualitative Methods in Empirical Studies of Software Engineering", IEEE Transactions on Software Engineering, 25(4):557-572, July/August 1999.

[24] Floyd J. Fowler Jr, "Survey Research Methods", 3rd edition, ISBN: 0761921915, Sage publications, 2001.