Development with Off-The-Shelf Components: 10 Facts

Jingyue Li, Reidar Conradi, Christian Bunse, Marco Torchiano, Odd Petter N. Slyngstad, Maurizio Morisio

jingyue@idi.ntnu.no, conradi@idi.ntnu.no, Christian.Bunse@i-u.de, marco.torchiano@polito.it, oslyngst@idi.ntnu.no, maurizio.morisio@polito.it

Abstract

The paper summarizes the results of several industrial surveys on issues related to the development of systems using Commercial-Off-The-Shelf and Open Source Software components. The results demonstrate the following. (1) There is a discrepancy between academic theory and industrial practices regarding the use of components. One reason is that researchers have empirically evaluated only a few theoretical methods; hence, industrial practitioners currently have no reason to adopt them. Another reason might be that researchers have specified the contexts of application of only a small number of theories in sufficient detail to avoid misleading users. (2) Academic researchers often hold false assumptions about industry. For example, research on requirement negotiations often assumes that a client will be interested in, and be capable of, discussing the technical details of a project. However, in practice this is usually not true. In addition, the quality of a component in the final system is often attributed solely to component quality before integration, ignoring quality improvements by integrators during component integration.

Keywords: COTS-based development, OSS-based development, empirical studies.

0. INTRODUCTION

A software component (henceforth, component) is a unit of code that integrators can combine with other components and integrate into a system in a predictable way. Software developers build components on the principle of "build once, reuse often". Hence, the use of components promises to reduce development time and cost while increasing software quality. An IDC survey in early 2007 illustrates that more than 50% software developers have used software components for development in the most recent projects [1].

Components from third parties (so called Off-The-Shelf (OTS) components) are of different types, i.e. Commercial-Off-The-Shelf (COTS) and Open Source Software (OSS), which makes composition a complicated task that requires risk-management techniques. In principle, OTS component-based development involves three stakeholders: component provider (i.e. COTS vendor or OSS community), application integrator, and application client. Different stakeholders face different issues and challenges. Application integrators must manage processes and knowledge well to ensure successful component selection, component integration, and component maintenance. To meet such goals, integrators need to communicate with component providers to get information and support. They also need to coordinate with clients to determine requirements as well as to get the OTS-based system accepted. Figure 1 summarizes software development with OTS components from the integrators' perspective.

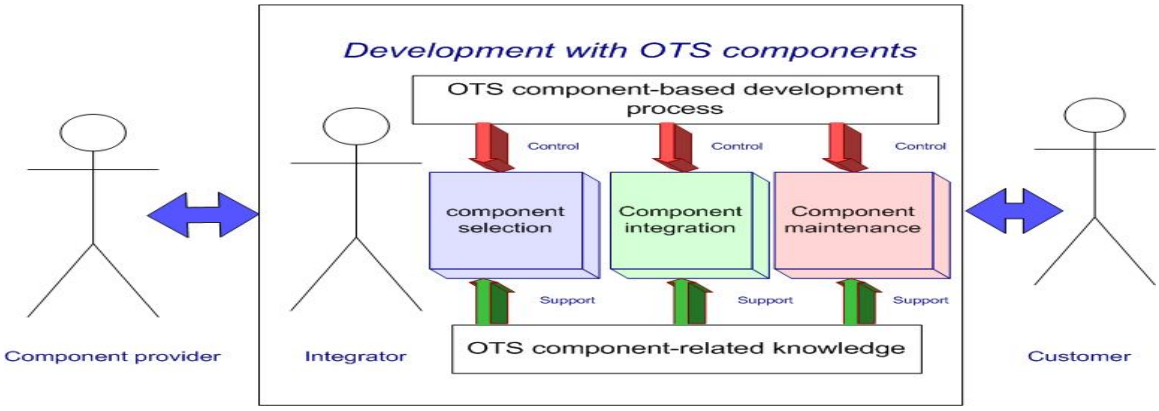

Figure 1. Development with OTS components: actors and activities

Researchers have proposed several methods for improving processes and managing risks to facilitate the integration of OTS components [2] [3]. However, they have evaluated few of these by industrial case studies [4]. *This makes it difficult for project managers to evaluate the effectiveness of proposed methods and to make the right decisions on the basis of empirical evidence.*

With these problems in mind, we performed a series of studies (see the side bar) to investigate the state of the practice in OTS-based development and the reasons for applying these practices. We here report the results of the last two steps of our studies: (i) an industrial survey with 133 completed projects from 127 companies, and (ii) 28 follow-up telephone interviews. Detailed information of the participated companies and projects is in [5].

- 1. Ten facts about industrial practices on OTS component-based development

Our findings illustrate that there are discrepancies between the proposals of academic researchers and industrial practice.

1.1 FACT 1

> ***Development process:*** *Companies use traditional processes enriched with OTS-specific activities to integrate OTS components.*

Boehm et al. regard neither the waterfall model nor the evolutionary development model as suitable for COTS-based development [2]. When using the waterfall model, integrators identify requirements at an early stage and choose COTS components at a later stage. This increases the likelihood that COTS components will not offer certain features that are required. Evolutionary development assumes that additional features can be added quickly if required. However, developers find it difficult to upgrade COTS components: the vendor will not change the product upon the request of a single client, and the absence of source code prevents the development team from adjusting or adapting the COTS components. This would suggest that companies need to adapt their development processes in response to using OTS components.

In fact, out of 75% (100/133) of the projects we investigated in the main study, developers chose their main development processes before they even started to think about using OTS-components.

Why do not companies adapt the development process? Four out of the 28 companies interviewed in the follow-up study are immature and have no well-documented development processes. They insisted on their ad-hoc development processes without considering any changes

due to the use of OTS components. They regarded the introduction of formal and heavyweight development techniques proposed in the literature as ***useless***, because OTS components either have been integrated several times in previous projects, or do not constitute important parts of the overall system. Surprisingly, five interviewees in the follow-up study thought that there was no difference between selecting/integrating OTS components and selecting/integrating software classes from an internal library. Two participants using agile processes believed that adaptation was not necessary, because agile means lightweight and depends solely on developers' experience.

Side bar: Genesis of the study

Our studies were inspired by a qualitative study [6] of COTS usage in seven IT companies in Norway and Italy, done in 2002. The study identified six "theses" on COTS usage, partly challenging commonly held beliefs. To build upon this study, we first conducted a qualitative **prestudy** *[7] in 2003, using a structured interview. We interviewed project managers of 16 projects from 13 Norwegian IT companies to summarize their lessons learned from COTS-based development. To verify the conclusions of our prestudy, we then developed a quantitative* **main study** *[5], which we performed as a survey in 2004 and 2005 to address IT companies in Norway, Italy and Germany. In this survey, we investigated the process improvement and risk management issues in 133 (47 from Norway, 48 from Germany, and 38 from Italy) completed COTS or OSS component-based projects that we selected using a stratified-random sampling strategy [8]. The response rate of this study shows that 53% of investigated software development companies had already completed OTS component-based projects by 2005. Results of this survey illustrate the state of the practice of OTS component-based development. To determine the reasons for phenomena discovered in the main study, we conducted a* **follow-up** *study, using telephone interviews with 28 participants (six in Norway, 12 in Germany, and 10 in Italy) selected by convenience.*

If companies adapt processes, how do they do so? They typically added a prototyping phase during OTS selection to evaluate and learn about OTS components. One of them uses the RCPEP process proposed in [9]. Two others are moving toward the new version of the V-model, i.e. V-model XT, which has been explicitly adapted for use with OTS components.

Our insights: Familiarity with the OTS component is proposed as a leading factor to be considered when selecting OTS component [6]. Our results show that the familiarity with OTS candidates is also an important factor to be evaluated for customizing the whole development process. Although companies are using adapted evolutionary (e.g. RCPEP) and waterfall (e.g. V-model XT) processes to integrate OTS components, sufficient knowledge with OTS candidates may make the usage of these adapted processes unnecessary.

1.2 FACT 2

Component selection: *Integrators select OTS components informally. They rarely use formal selection procedures.*

Researchers have proposed several procedures for formally selecting COTS components that promise 'fail-safe' decisions (see, for example, [3] for a summary). However, when we analysed the data from our main study, we found that, in practice, integrators habitually select components in an ad-hoc manner, using in-house expertise and/or web-based search engines.

Why do not companies use formal selection processes? We found that in some cases, integrators had simply neglected steps for searching and evaluating OTS components. The interviewees gave two reasons: (i) only a limited number of OTS candidates were available in the market, and (ii) the integrators' company already had a long-term partnership with a specific provider. However, about 20% of interviewees in the follow-up study were unaware of the formal selection processes proposed by academia. Most interviewees were sceptical about the cost-effectiveness of using a formal process, especially under time-to-market pressure. They would rather trust the experience of in-house expertise than any formal process, as discovered in [6]. In agile projects, integrators did not consider formal OTS selection processes at all, because they believed that adopting any kind of formal procedure would undermine the agile nature of the development process.

If a formal selection process was applied, what was done? Only one of the 28 interviewees stated that his company had used a formal process for selecting OTS components. The project on which this interviewee worked was safety- and performance-critical. A candidate component had to fulfil several quality requirements and had to follow strict industrial standards. Integrators adopted candidates on the basis of either client recommendation or a search of other sources. They then narrowed down the number of candidates by reading the literature and scanning discussion boards. After that, they purchased and installed the remaining candidates and used them in a small prototype project, in order to evaluate both non-functional properties and functionalities against requirements. Integrators also evaluated the components' compliance with the given industrial standard.

Our insights: Researchers have evaluated few formal processes for selecting OTS components empirically, with the aims of measuring their cost-effectiveness and determining their applicability in certain contexts. Thus, the pre-conditions and benefits of using a formal process are unclear. Without evidence on the possible benefits, integrators are reluctant to use formal processes, which are supposed to be complex and time-consuming.

1.3 FACT 3

> ***Component selection:*** *There is no specific phase of the development process in which integrators select OTS components. Selecting components in early phases has both benefits and challenges.*

Researchers usually suggest that integrators select components in an early phase of development, so that they can identify possible problems early. The data from our main study showed that most integrators did select OTS components in the early phases of a project, e.g. prestudy (38%), requirement specification (30%), and overall design (16%). However, some integrators selected OTS components in later phases, i.e. detailed design (6%), and even coding (7%). (Three percent of participants did not know when the component was selected).

Reasons for, and issues pertaining to, selecting OTS components in the prestudy phase: One reason is that the component will drive the definition of the architecture of the whole system, as discovered in [6]. Several interviewees had bad experiences of system architecture restructuring when they had selected components in later phases. Furthermore, integrators often decide to (re)use familiar components in the prestudy phase. However, some interviewees recognized that if they select a component from a set of unknown OTS components in such an early phase, they must use comprehensive documentation and the results of trials. Yet documentation is often absent and, when it exists, often does not describe the actual component accurately. This, in turn, requires unexpected extra effort in later phases. Moreover, at the beginning of a project, integrators do not know all the required functions of a system. Thus, they may have to write adapters or change OTS components later.

Reasons for, and issues pertaining to, selecting OTS components in the requirements or design phase: Most integrators selected OTS components during the requirements or overall design phase. The interviewees identified benefits of selecting OTS components in these phases as follows:

- The integrators know the system architecture and the functional requirements for a possible component. They thus have a solid basis on selecting OTS components.
- Integrators can readily adapt the system to the specific needs of the component, thus enabling integration to proceed seamlessly.
- Once the integrators have defined the architecture and have selected the OTS candidates, they can easily define test-cases and make systematic plans for quality assurance.
- Once the integrators have defined the requirements and have selected the OTS candidates, they can estimate the cost of the project more accurately.

However, selecting OTS components in these phases carries certain recognized risks and challenges, as follows:

- Integrators usually do not care enough about technical details in an early stage. This may subsequently lead to problems of implementation and integration.
- During the course of a long project, the providers of an OTS component may release a new version during the detailed design or coding phases. Consequently, the integrators may need to re-evaluate the component and redesign the system.
- In projects using agile development processes, integrators typically identify and document requirements by means of "user stories", and set up the entire process in such a way that they can make changes easily. Therefore, the earliest possible phases at which integrators should think about components are either the detailed architectural design or the development iterations. However, this may require integrators to expend extra effort on refactoring the system.

Our insights: Most approaches assume that selecting components in the early phases of a project will yield benefits [10]. However, we have identified pitfalls that integrators must consider if they wish to select OTS components in the early phases of a project.

1.4 FACT 4

> ***Component integration:*** *Estimators use personal experience when they estimate the effort required to integrate components and most of the time they do not estimate accurately. Stakeholder-related factors will affect dramatically the accuracy of estimates.*

In 83% of the 133 projects that we examined in the main study, the estimations of the effort required for integration were unsatisfactorily estimated. Only four out of the 28 interviewees used a formal effort estimation tool, e.g. COCOTS [11]. The remaining interviewees estimated the integration effort solely on the basis of personal experience.

Reasons for inaccurate effort estimation. In addition to the usual factors that may affect the accuracy of effort estimation, the following factors also contributed to inaccurate effort estimations of projects examined in the follow-up study:

- It takes time to understand how to use the components correctly, because technical details of OTS components are not described explicitly in their documentation.
- The clients changed their requirements significantly. It is difficult to satisfy changed requirements due to the inflexibility of OTS components.

- The OTS provider did not respond quickly to required changes. Integrators waste a lot of time waiting for the providers to respond.
- The OTS provider released new versions of OTS components during the project. The integrators then had to expend extra effort on adapting the system to the new versions or on evaluating and integrating them.

Our insight: Some estimation tools, e.g. COCOTS [11], take into account both the technical nature of the components and a number of issues mentioned above, e.g. component understandability and vendor response time. Our data show that estimation tools should also take into account possible changes in requirements and the evolution of components, especially for large projects with long durations.

1.5 FACT 5

> ***Quality of the integrated system:*** *Negative effects of OTS components on the quality of the overall system are rare.*

The quality of OTS components is expected to be at least as good as that of in-house built components [12]. Our data support these expectations. The traditional qualities (reliability, performance, and security) of OTS components were a problem in the final system for few of the projects that we investigated.

Reasons for positive feedback on the quality of OTS components. Some interviewees in the follow-up study stated that their system was of high quality because the integrators evaluated and tested the OTS components carefully in the selection phase. Others stated that their experience with specific OTS components and the strategy of only using mature OTS components were helpful. However, another very important reason for the integrator's positive feedback on the quality of OTS components is that their expectations are not very high, either because the OTS components play only a minor role in the composed system, or because the integrators accept "minor problems" with free or low-cost components.

Our insight: In traditional software development, the quality of software is measured by how well it satisfies the client's requirements. For OTS components, there are two clients: a direct client (i.e. the application client) and an indirect client (i.e. the application integrator). When measuring the quality of components, people still refer to how well the component satisfies application client requirements after it has been integrated [12]. Our findings illustrate that, for various reasons, for example, low cost, application integrators sometimes accept OTS components that are of less than perfect quality. It is the integrator's quality assurance effort during selection and integration that ensures the quality of the OTS component in the final system.

1.6 FACT 6

> ***OSS and COTS components:*** *Integrators usually used OSS components in the same way as commercial components, i.e. without modification.*

People often assume that the commercial vendors of COTS components sell a copyright license with agreed specific support and do not make the source code available, while the open source communities that provide OSS components offer freely accessible source code yet promise no specific support. The study [6] illustrates that COTS component debates should include open source component. Results of our study supported observations of [6] and found that one third of the companies we investigated in the main study do have access to the source code of COTS components. However, only 15% of the COTS component integrators and 36% of the OSS component integrators changed the source code. In most cases, they used the OTS component "as-is".

Reasons for changing the source code in the projects investigated in the follow-up study. Integrators have to wait too long for providers to update their components.

Reasons for not changing the source code in the projects investigated in the follow-up study:

- Source code is unavailable.
- The fast and effective OTS component support renders change unnecessary.
- Developers lack the deep knowledge and thorough documentation that is required to change the source code.
- It is difficult to get changes accepted into OTS component in later releases. Integrators do not want to change the source code, so that they can "drop-in" replacements when the next version of component is released. In-house changes run the risk that the system will be incompatible with later component updates and that maintenance will become too difficult.
- Changing source code may generate legal issues. For example, the integrators have to take responsibility for any problems caused by the changed components.

Our insights: Changing the source code of OSS components may not be feasible, especially for a long-term commercial system with a possibly long evolution path ahead. Thus, application contexts, e.g. commercial vs. non-commercial application and long-term vs. short-term application need to be considered when deciding to use OSS or COTS components.

1.7 FACT 7

> ***Locating defects is difficult:*** *Although problems with OTS components are rare, the cost of locating (i.e. within or outside OTS components) and debugging defects in OTS-based systems is substantial.*

Although integrators are, in general, satisfied with the quality of OTS components (see fact 5), in 80% of the projects that we investigated integrators experienced difficulty in locating defects when they occurred.

Reasons for inefficient defect location: The following factors can cause failures within an OTS-based system: defects in OTS components, misuse, or defects in the code to integrate OTS components with other parts of the system (besides defects in the subsystems built in-house). If the documentation is incomplete or imprecise, or the source code is inaccessible, unfamiliar, insufficiently commented, or messy, application integrators find it difficult to locate the defects by themselves. Asking the provider for help may create new problems. Component providers are usually reluctant to read the code from application integrators to locate defects; especially when components from different providers are mixed. One interviewee tried test-driven development to mitigate this problem. He was unsuccessful because it is difficult to write test cases for an OSS component without in-depth knowledge of its code.

Our insight: The variety of deployment environment and configuration of OTS components hinders OTS providers to reproduce the reported errors. The irreproducible errors usually will not be prioritized and fixed by OTS providers. To improve the debugging efficiency of OTS-based systems, integrators need to work collectively with OTS providers by engaging in a process of constructing a context where the OTS provider can reproduce the reported error [13]. Moreover, integrators need to investigate relevant descriptions with the reported error from mailing list, web forums, and bulletin board in order to provide more information to and convince OTS provider that the error may potentially affect many systems.

1.8 FACT 8

> ***Relationship with the provider:*** *The relationship with the OTS component provider involves much more than defect fixing during the maintenance phase.*

Researchers have claimed that a good relationship with the provider is essential for a successful OTS component-based project [14]. However, most previous studies emphasize only the technical support from providers in the maintenance phase, e.g. fixing defects. Data from our study show that additional issues related to providers need to be considered.

Issues related to component providers. In the *component selection* phase, integrators need to both evaluate component candidates and to evaluate and consult with their providers. This evaluation includes not only the reputation of a provider, but also such matters as technical support, market share, and company size. Several interviewees stated that, in their experience, OSS communities that have large user groups usually provide better support than those with smaller ones. In addition, the integrator and provider must consider legal and licensing issues and specify them in the contract. Typical example are the response times of COTS vendors (e.g. on problem reports) and the responsibility for returning code changes of OSS components. In the *component integration* phase, OTS providers need to be contacted to ease debugging, to provide extra functionalities, and to fix defects. Integrators believe that knowing a specific person at the COTS vendor company or in the OSS community is essential for reducing integration costs. The readability, accuracy, and completeness of the component documentation made available by the provider also affect the efficiency of integration. In the *maintenance phase*, integrators need the provider's help for debugging defects and for suggestions/hints on component evolution or on reusing the component in the future.

Our insights: Different people in OSS communities may be involved in different tasks to support the use of a component. For example, some senior OSS community members, who have better views on the similarities and differences between their components and others, may help OSS users to make the right evaluation and selection. Other community members, who have solid experience of fixing bugs and customizing software features, may help to ease integration. It is therefore important for integrators to know the right persons for a specific task in an OSS project, to share detailed experiences with them regularly, and to build partnership with them [4]. Hence, OSS projects need to specify contact persons on the basis of possible user needs.

1.9 FACT 9

> ***Relationship with the client:*** *Involving clients in OTS component decisions is rare and sometimes unfeasible*

OTS components seldom satisfy all of a client's requirements; hence, researchers regard the (re)negotiation of requirements with the client as an important strategy in OTS-based development. However, our data from the main study show that integrators rarely involve clients in the "build vs. acquire" decision, or in selecting OTS components.

Reasons for not involving clients: Half of the companies that we investigated in the follow-up study are software houses, which produce software for the general market. Thus, they have no direct client with whom to confer when developing their products. The others develop software for a dedicated client, but most of their clients have either no interest in discussing, or insufficient technical capabilities to discuss, such issues. Only two out of 28 interviewees had involved their clients in discussing the outcome of selecting OTS components. In one project, the client was mainly interested issues pertaining to reselling and licensing, cost, or compliance with given industry standards. In the other project, the client had to be involved because the project included

cooperative and distributed development. The client had to centralize OTS component selection to ensure that all the subcontractors could understand and use the selected OTS components.

Our insight: Application clients usually only care about the final products and typically are not interested in the technical details of the implementation. Most approaches to the (re)negotiation of requirements that researchers have proposed simply assume that application clients have enough background competence to discuss technical details [14]. We encourage companies to clarify, at the start of the project, the clients' interests and technical capabilities so that they can decide on possible strategies for (re)negotiation.

1.10 FACT 10

> ***Knowledge management:*** *Knowledge that goes beyond the functional features of OTS components must be managed.*

The success of OTS-based development projects requires that companies manage the implicit and explicit knowledge about OTS components. More than half of the companies that we investigated in the main study already have dedicated staff (so-called "component uncles") to keep the OTS component-related knowledge. Most of them are experienced software architects and senior developers.

Which knowledge needs to be kept and shared? (1) Companies need to capture knowledge about a component itself, e.g. basic functionality, standards conformance, side-effects, undocumented issues, and non-functional properties. (2) Companies need to manage knowledge about how to facilitate component integration: licensing and reselling obligations, examples of the code that is used to connect the OTS component to the system (gluecode), and descriptions of possibilities for optimization. (3) Companies need to acquire and store information about the stakeholders. This will include client preferences, with whom to negotiate at the client side, whom to contact at the provider side, and who knows which components at the integrators' organization.

Which knowledge management mechanisms to choose? The software market changes quickly and OTS components have short release cycles. Hence, the knowledge that developers need to acquire and share will change quickly as well. In order to accommodate this changing demand, some of the interviewees advocated storing and sharing tacit knowledge through personal communications, e.g. coffee-meetings, internal seminars, informal discussion forums, or regular meetings in (agile) development groups. Other interviewees preferred more formal and recordable approaches to mitigate the problem of losing experience when a key person leaves. Some of them even claim that every project should have a touchdown meeting where they can share their collective experience. Several of the companies that we investigated in our follow-up study have set up a small Wiki site to share knowledge. Companies have also used a central authority (i.e. an OTS team or "component uncle") to manage OTS-related knowledge and yellow pages to record "who knows what". Integrators regard these as effective mechanisms for managing knowledge.

Our insight: Our results show that implicit and explicit knowledge about OTS components has been partly managed within the organization by "component uncles". However, there are very few centralized external channels for OTS users to share and communicate experience between organizations. External experiences of using certain OTS components are scattered in several COTS or OSS portals, bulletin boards, or mailing lists. Searches in search engines usually yield huge, unwieldy sets of results. A centralized experience portal for sharing OTS component-related knowledge between organizations, probably using a global OTS Wiki [15] could be a solution.

2. Conclusion

The results of our industrial surveys have revealed gaps between theory and practice regarding the use of OTS components. We suggest that researchers need to be more precise about the assumptions and contexts of the application of their proposals regarding the revision of OTS-based development processes, the processes by which companies should select their components, and the processes by which integrators and providers negotiate requirements. We further suggest that researchers conduct more empirical case studies, to investigate cost-effectiveness of proposed theories. We suggest integrator to collaborate more actively with OTS providers to facilitate debugging the defect. We also suggest integrators to investigate the strategies for (re)negotiating requirements with clients at the early stage of the OTS-based project.

Our surveys also reveal several issues that researchers need to address. By what means can providers and integrators share knowledge of OTS components on a global scale? How can people working on the field establish the "who to contact" yellow pages for each OSS project, to facilitate support from OSS communities?

As an important caveat, note that we have, thus far, collected only a small amount of data. We were the first to perform such an empirical study using a random sample of IT companies. Researchers need to perform further studies, both to validate our results and to align them with the latest progress in the field.

3. Reference

[1] http://www.idc.com/getdoc.jsp?containerId=IDC_P644, 2007.

[2] B. Boehm et al., "COTS integration: Plug and Pray?" *Computer*, vol. 32, no. 1, 1999, pp. 135-138.

[3] K. RPH. Leung et al., "On the Efficiency of Domain-Based COTS Product Selection Method," *J. Information and Software Technology*, vol. 44, no. 12, 2002, pp.703-715.

[4] J. Norris, "Mission-Critical Development with Open Source Software: Lessons Learned," *Software*, vol. 21, no. 1, 2004, pp. 2-9.

[5] J. Li et al., "Validation of New Theses on OTS-Based Development," *Proc. 11th Int'l Symp. Software Metrics*, IEEE, 2005, pp. 26.

[6] M. Torchiano et al., "Overlooked Facts on COTS-based Development," *Software*, vol. 21, no. 2, 2004, pp. 88-93.

[7] J. Li et al., "An Empirical Study of Variations in COTS-based Software Development Processes in Norwegian IT Industry," *J. Empirical Software Engineering*, vol. 11, no. 3, 2006, pp. 433-461.

[8] R. Conradi et al., "Reflections on conducting an international CBSE survey in ICT industry," *Proc. 4th Int'l Symp. Empirical Software Engineering*, IEEE, 2005, pp. 214-223.

[9] P. K. Lawlis et al., "A Formal Process for Evaluating COTS Software Products," *Computer*, vol. 34, no. 5, 2001, pp. 58-63.

[10] I. Crnkovic et al., "Component-based development process and component lifecycle," *Proc. 27th Int'l Conf. Information Technology Interface*, IEEE, 2005, pp. 591-596.

[11] C. Abts et al., "COCOTS: A COTS Software Integration Cost Model - Model Overview and Preliminary Data Findings," *Proc. 11th ESCOM Conf.*, 2000, pp. 325- 333.

[12] J. M. Voas, "Certifying Off-the-shelf Software Components," *Computer*, vol. 31, no. 6, 1998, pp. 53-59.

[13] Ø. Thomas et al., "Debugging Integrated Systems, an Ethnographic Study of Debugging Practice," *Proc. 23rd Int'l Conf. Software Maintenance*, IEEE Press, 2007.

[14] L. C. Rose, "Risk Management of COTS Based Systems Development," Springer LNCS, vol. 2693, 2003, pp. 353-373.

[15] C. Ayala et al., "Open Source Collaboration for Fostering Off-The-Shelf Components Selection," Proc. 3rd Int'l Conf. Open Source Systems, Springer, 2007, pp. 17-30.