Internal Marketplace as a Mechanism for Promoting Software Reuse

Maria Ripatti[1], Terhi Kilamo[2], Karri-Tuomas Salli[1] and Tommi Mikkonen[2]

[1]Insta Defsec Ltd, Sarankulmankatu 20, FI-33901 Tampere, Finland
maria.ripatti@insta.fi, karri-tuomas.salli@insta.fi
[2]Department of Pervasive Computing, Tampere University of Technology,
Korkeakoulunkatu 1, FI-33720 Tampere, Finland
terhi.kilamo@tut.fi, tommi.mikkonen@tut.fi

Abstract. Reuse is one of the classic ways to improve productivity in software development. Indeed, benefiting from software components, patterns, and solutions that have been developed in the company potentially leads to savings in all phases of software intensive work. However, putting such an approach to practice is far from being simple. In particular, when considering software companies that specialize in customer-specific software projects, it is common that similar designs and technology choices are made in parallel without project-crossing knowledge. In such settings, there is a lack of a systematic approach between projects to spread good practices or to eliminate bad ones. In this paper, we propose solving such problems with an information system that acts as a marketplace for promoting software reuse within a project organization, much to the same flavor as app stores are used to promote mobile applications. The paper provides insight to the design of our prototype system, as well as contains preliminary views from users in one organization.

Keywords: Reuse, software projects, inner source, marketplace.

1 Introduction

The business climate today is highly competitive for software companies. This leads to a constant need to look for improvement in development processes in order to maintain the competitive edge. One way to achieve this is reuse – benefiting from software components, patterns, and solutions priorly developed in the company [1]. The benefits reuse promises are improvement in software quality, performance, and reliability [2, 3]. When implemented well, code reuse can shorten development time, which in turn shortens time to market. It can also help to avoid redundant work in projects, such as analysis phase, thus improving project productivity and reducing development effort [3]. Reuse can also make software maintenance easier as reuse unifies coding practices between projects [4]. All in all, the promise of reuse is undisputable.

Project organizations still largely carry out their businesses in accordance to the old subcontracting model, where the customer defines requirements and the

project organization is optimized to perform technical activities needed for requirements elicitation, design, and final implementation. In this context, where each project is treated as a separate entity due to e.g. confidentiality reasons, spreading word regarding successful technology choices gets overly complex. Each technology selection will strictly remain in the project silo instead, and experiences regarding using them are only reused once the developers are allocated to future projects. Therefore, the developers are constantly faced with a challenge of finding suitable solutions to current programming tasks. This includes being constantly on the lookout for components to reuse as such are not readily visible across project barriers. Furthermore, and even more counterproductively, they end up assiduously solving the same problems over again. The challenge posed hence lies in making the components suitable for reuse visible for the developers across projects and over silo borders.

Commonly used approaches to implement reuse include product-line architectures [5] and inner source [6], both of which introduce established reuse processes. Product-line architectures provide a common platform for a typically domain-specific family of products. Each product is then developed by adding the product specific features on top of the shared platform. Inner source, sometimes coined internal open source or corporate source [7], sometimes progressive open source [8], in turn refers to the practise of utilizing suitable open source software development practices and tools within an organization. In general, both approaches build on creating software repositories [9] as a method to give developers access to reusable components. Still, finding suitable components from a large and typically constantly growing repository is challenging [10]. Additionally the lack of decent documentation makes it difficult to evaluate how suitable a component is for reuse. Furthermore, the amount of work required for using it as a part of another piece of software can be hard to estimate. In this paper, we propose an approach to turn this around much like modern online distribution platforms for mobile applications, colloquially app stores, have simplified installing compelling applications that were impossible to find before they were made available through an information system that also syndicates users' views.

To summarize, while the idea of reuse is decades old and different approaches to classifying and representing reusable components in repositories have been around for a while [10, 11] reuse is still not a fluent everyday practice at software companies. This paper addresses the question: how should reuse be implemented within a project organization in order to avoid the problems and challenges repositories bring forth?

The paper presents a pilot study on establishing a company internal component marketplace that engages the ideology of app stores in order to highlight fitting components to developers for reuse across in-house projects. The paper further presents the systematic reuse process alongside and applied to the marketplace. As our prototype, such a marketplace was implemented and taken into use in a mid-sized Finnish software company developing a range of software products mainly to large customers that require confidentiality, high quality and predictable delivery. In short, the paper contributes:

– the concept of using a component marketplace to promote reuse in an organization,
– an industry scale prototype implementation, and
– the reusability process adopted in conjunction with the marketplace.

Here, we address the very first views to deploying using such a system in a project organization. A detailed case study regarding the experiences will be reported later in a separate article, where the resulting increase in reuse will be evaluated.

The rest of the paper is structured as follows. Section 2 presents background on the business environment of the project organization, the challenges of software reuse and discusses the requirements in the adoption of systematic reuse. Section 3 describes the concept of a marketplace as a mechanism for supporting component reuse. Section 4 describes how requirements were gathered from the target organization. Section 5 presents the implemented marketplace solution and discusses how it takes the main challenges into account. Section 6 describes the reuse process adopted at the company. Section 7 discusses the key findings of the study. Finally, Section 8 concludes the paper with future research directions.

2 Background

The benefits of reuse – better quality and reliability, shorter development time, and unified practices – are so overwhelming, that it almost beyond understanding why so many organizations overlook this opportunity. Then again, when considered from the viewpoint of a software company, it is usually self-evident that a lot of effort must be invested in creating practices needed for systematic reuse. The enticing prospect of reuse can and often is hindered by the intimidating challenges in making it a successful ongoing process – if the processes of reuse are not planned and well-established the organization may end up in a situation where a better and faster solution would be to just implement each project on its own. One must be able to identify the reusable components as reuse is not a fit-for-all solution [12]. The components intended for reuse need to be generalized and documented properly [13]. Reuse itself requires finding the suitable components, getting to know them and making possible changes to them [4]. As a concept, reuse has been around from the sixties [14] – and we are still struggling with it.

2.1 Towards Software Reuse

The best results can be obtained with systematic reuse [15]. This also requires that the challenges and problems that reuse entails are identified in the project organization. Even the best laid plans do not alone guarantee a successful adoption of reuse. It needs to be encouraged as a best practice on an organizational level as well as supported by development practices and infrastructure.

Jacobson et al.[13] propose four key processes that are needed in successful reuse (see Figure 1).

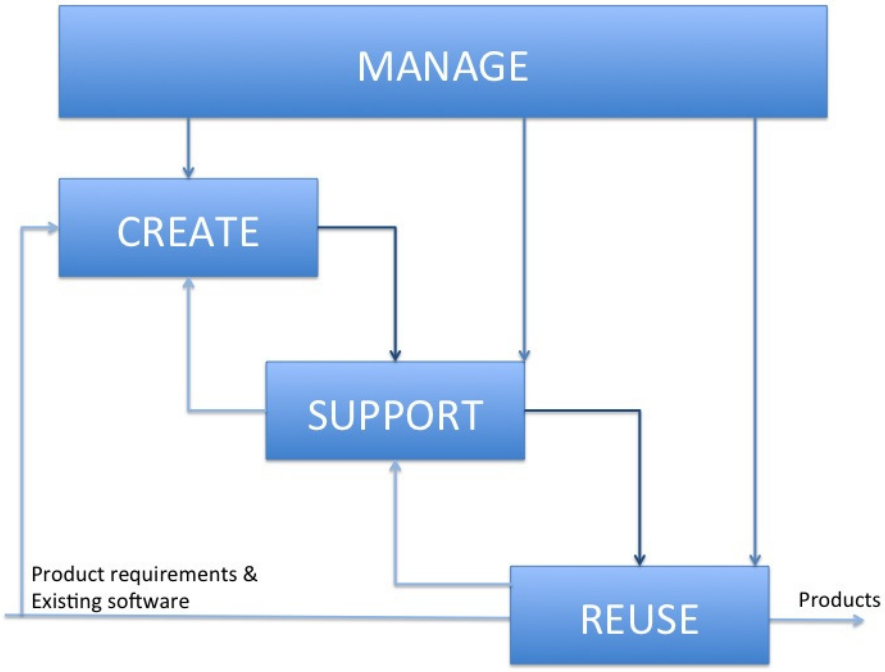

Fig. 1. Four concurrent processes of reuse [13].

- **Create**: The creation process focuses on identifying the needs of and provide for the projects. The development activities included comprise of numerous issues, such as domain planning, development and installation environment, selecting components that are to be reused, and tools that are used for reuse. In many cases, these need a lot of attention in order to introduce generic solutions instead of context-specific ones.
- **Support**: Supporting reuse includes numerous human-related issues. For instance, components to be reused need to be classified and packaged in order to support reuse; distributing them needs common practices; and instructions need to be in place to make all this happen in accordance to plans. The focus is on supporting the needed processes and maintaining the component collection.
- **Manage**: Leadership functions – including planning, funding, resourcing, prioritization, coordination, as well as many other leadership related functions – is often the main obstacle when considering reuse. Balancing between the long-term investment in coordinated reuse and everyday needs of going forward with projects is difficult. Moreover, many of the present agile software engineering approaches, such as Scrum [16] and Kanban [17], are often understood as focusing on satisfying customer requirements. They do not emphasize issues that will help the company in the long run.

– **Reuse**: Actual reuse takes place by selecting the components to be reused, which is a key characteristic for aiming at systematic reuse. Furthermore, systematic reuse entails customizing and combining the reusable components.

Putting all this to practice in a software organization requires management attention, organizational engineering, and hence time that is away from productive work. That in turn is the key ingredient for creating revenues in software companies who specialize in delivering software through customer-specific projects. There, it is common that problems hindering reuse emerge. These include considering only reusing individual components, overly generic designs, lacking scalability, legacy technology, and reuse only for reuse itself [9]. Boehm further emphasizes the danger of "the field of dreams": believing that builiding a software repository of reusable components suffices to make reuse an everyday practice.

The open source software movement [18] with its development ideology and community-driven approach [19] has risen as a force to be reckoned with when it comes to developing high-quality software products. The concept of a software forge [20] –an repository of projects that can be browsed and that provides the necessary development tools – comes from open source. In its wake, inner source – internal open source – has been utilized by companies to adopt the methodology and ideas of open source software development within a software company. As such, inner source on the development infrastructure level provides a plausible platform for reuse [21]. Inner source does however also pose its set of challenges [21] such as identifying the suitable components and selecting them based on evaluation, poor documentation, as well as integration and architecture issues. These as such are identifiable further as the challenges of reuse itself. However, the factors supporting adoption of inner source [22] – the idea of a seed software product, practices and tools, and organizational and the community-driven approach can also be seen as key factors in supporting reuse.

2.2 Context

The company where the experiment for the proposed approach is carried out, Insta DefSec Ltd[1], is a Finnish software organization that specializes in developing a range of company software projects. The company delivers software mainly to large customers that require confidentiality, high quality, and predictable delivery time. Their potential customer-base includes such governmental organizations as the defence forces and other organizations that often require a certain level of independence of other projects in their execution largely due to their confidential nature. In fact, each project may have different requirements on levels of confidentiality, which may have an effect on the personnel that may participate in them. Despite these limitations and project boundaries systematic, well-organized reuse is beneficial also to the customers. The major programs affect the

[1] `http://www.insta.fi/en/`

company line structure, for example, due to the requirements and project restrictions. Each program usually holds and manages its own resources, including software engineers working on the project as well as almost all the technical project data. Consequently, information regarding successful or unsuccessful technology decisions or design practices is mostly distributed across project borders through word-of-mouth from developer to developer, which though effective in breaking the project silo in separate cases lacks in organization wide governance. For instance, if in one project, the developers perform an analysis regarding the automated mapping from an object-oriented design to a relational database, the results of the analysis would be shareable across all in-house projects. Moreover, experiences from using a certain application-independent component – be it developed internally or a third party system – could be beneficial across the projects, as many of them deal with similar technologies simply due to domain requirements. Without a systematic approach the benefits, such as joint maintenance of common technology, are not gained. The company is constantly looking for ways to improve its ways of working. Thus working towards new methods for systematic reuse and organization wide dissemination of best practices is one area where new directions are tried out.

3 Marketplace as a mechanism for reuse

Despite the promise of reuse code repositories do not seem to provide a sufficient solution to reuse. Instead they seem to bring issues similar to the challenges of reuse itself to the organization. Aiming for systematic reuse is not hence solved by setting up a code repository for the reusable components.

The goal of a company internal marketplace is to combine the idea of a digital online distribution platforms known from mobile apps such as Google Play[2] and Apple's App Store[3] – colloquially coined simply app stores – to these traditional reuse approaches in order to meet the challenges of reuse. This should increase the amount of reuse within the company and help to lower the boundaries between projects. The marketplace aims to solve the challenge of projects acting as knowledge silos. It is used to market components developed with the organization as well as third party components across projects thus taking them into use more straightforward for the projects. The app store features of the marketplace should further make locating of reusable component easier. The marketplace supports making a decision on reuse with clear descriptions on the available components as well as developer comments and instructions of use for them – all features familiar from the app store markets.

Philosophically, the marketplace incorporates the feel of inner source. It promotes transparency, acts as the "seed" product, and attracts contributions across the organization [22]. A component shared through the marketplace can, in addition to the components developed in the projects, also be a third party solution that could be valuable for several in-house projects. For example, in addition to

[2] `https://play.google.com/store/apps`
[3] `https://itunes.apple.com/app/apple-store/`

traditional software components or applications, the reusable component can be a reference implementation, a description of, or a design decision on, an architecture. A software library is shareable through the marketplace as well.

The marketplace can also decrease the effort required to reuse the components by offering the components directly in the marketplace. For reuse, the most important factor is that the marketplace offers an easy way to locate, add and describe components. Hence the components can physically be located outside the marketplace in a separate repository or online as long as the location is explicitly shareable through the marketplace. As summary, in order to meet the needs of systematic, successful reuse the marketplace should:

- act as an internal information channel for the organization
- make adopting reusable components more fluent
- guide development of reusable components
- guide technology choices made in projects
- motivate development toward reusable components
- help keeping track of available reusable components.

The marketplace aims to motivate further development of reusable component by enabling advertizing them through the marketplace [23]. Through the marketplace the projects can also give component recommendations. Finally, open source components are also shareable at the marketplace which allows the organization to better keep track of them. This way they can also ensure that the licence terms are known by the developers.

4 Requirements Gathering

Requirements elicitation was done in two stages to ensure that the internal marketplace would address the true needs of the developers. In the first stage, a rough list of requirements were gathered in meetings among a small group of people taking part in the marketplace research. Based on the requirements recognized there and from the conversations in the meetings, an inquiry about software reuse was created. The purpose of the inquiry was to identify more specific requirements.

In the second stage of the requirements gathering, the drafted inquiry was sent to the project managers, architects and developers of the company. The results confirmed that the requirements identified by the marketplace research team were accurate. The results also defined the priority of the requirements.

According to the results, the marketplace should enable sharing the following information regarding reusable components:

- basic information such as the name and version of components and other useful details
- the technical and functional descriptions,
- the locations and contact persons of the components, and
- prices and licences, if a 3rd party component was included.

Respondents also considered important that jar packages can be uploaded directly through the marketplace. If a direct marketplace access is not possible, it should offer direct links to repository locations of the components and any other important files such as license conditions, version history or bug databases. Respondents also pointed out that the marketplace can not be implemented as a cloud service because of the information security reasons. The gathered requirements were used in evaluating the possible marketplace implementations.

5 Marketplace implementation

The goal of deploying the marketplace within the organization is for it to support the systematic reuse process. Hence, the implementation should simplify the reuse process as a whole, including all the activities listed above, but with a particular focus on managing the reuse. Additional focus is put on promoting people to reuse both software assets as well as research work invested in selecting best possible third party libraries as these were the missing link between ad-hoc reuse and systematic reuse processes. As a solution, an information system was introduced. The system would gather all the necessary data into one, similarly to the digital online distribution platforms — app stores — that have become common in the mobile domain and in online stores.

Since several app and web stores exist already, we next performed an evaluation regarding the already existing implementations. As a result of the evaluation process, OpenCart[4] was chosen to the pilot use of the internal marketplace. OpenCart is free open source e-commerce platform for online merchants. OpenCart provides a professional and reliable foundation from which to build a successful online store. This foundation appeals to a wide variety of users; ranging from seasoned web developers looking for a user-friendly interface to use, to shop owners just launching their business online for the first time. OpenCart has an extensive amount of features that gives you a strong hold over the customization of your store.

OpenCart offers an e-commerce platform and admin portal. The e-commerce contains for example a front page, category pages, product pages, a shopping cart, a product comparison and the search of products. The front page is used for advertising products. Products are introduced at the category, product and comparison pages and they can be bought via shopping cart. The admin portal provides user and product management functionalities, reports about the e-commerce usage and the customization of the store.

Transforming OpenCart to a software component marketplace required some changes to both the e-commerce and the admin portal. OpenCart is primarily intended to selling tangible products and by default it does not support sharing electrical content. Due to this the product descriptions and the language used required customization. Furthermore, the e-commerce and admin portal contained many features and pages that were unnecessary for the marketplace. For

[4] `http://www.opencart.com`

example, the shopping cart, payment features and unused reports were removed. These changes were made directly to the source code.

As shown in Figure 2, the OpenCart design is based on the MVC design pattern and the language specific information is separated from the other content. The OpenCart directory structure contains separate folders for e-commerce and admin model, view, controller and language files. E-commerce files can be found in the catalog folder and admin files in the admin folder. The OpenCart installation contains also other folders such as system and image folder that contains classes that are used by both the e-commerce and admin portals. The structure of the OpenCart and the developing process of modules are extensively explained in the OpenCart documentation, which makes creating new modules and customization existing modules easy.

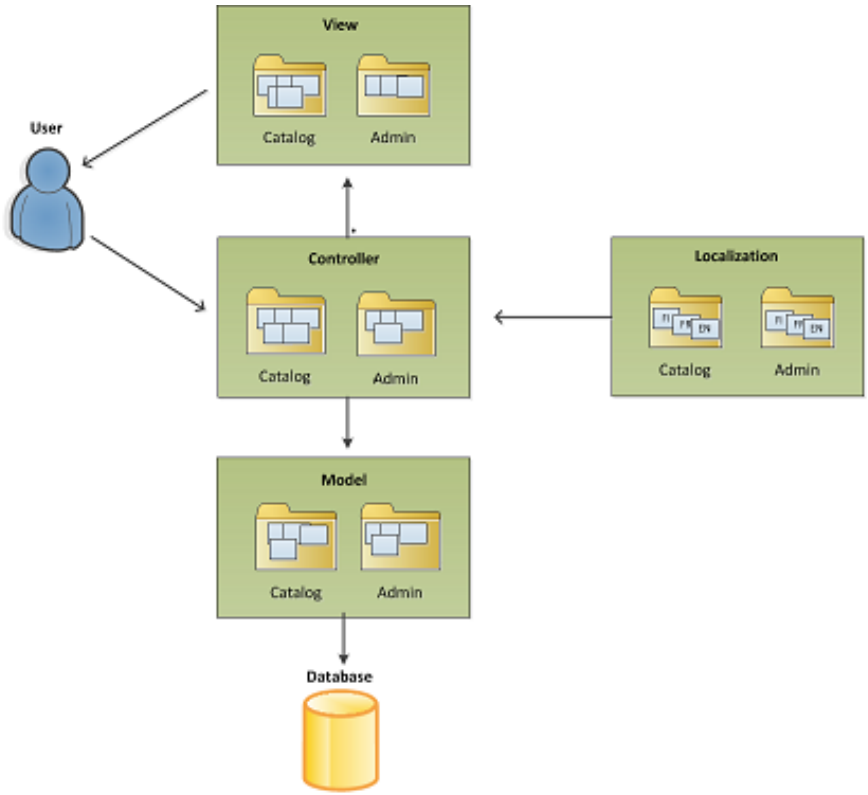

Fig. 2. OpenCart architecture.

The marketplace is used for promoting reusable components but actual components are located in the reuse repository because of the absence of electronical sharing. The modified marketplace (Figure 3) contains the front page that is

used for advertising reusable software components. The content of this page can be customized using the admin portal. In the front page given in the figure, there is a welcoming message, links to two components, as well as references to some topic areas according to which components can be grouped.

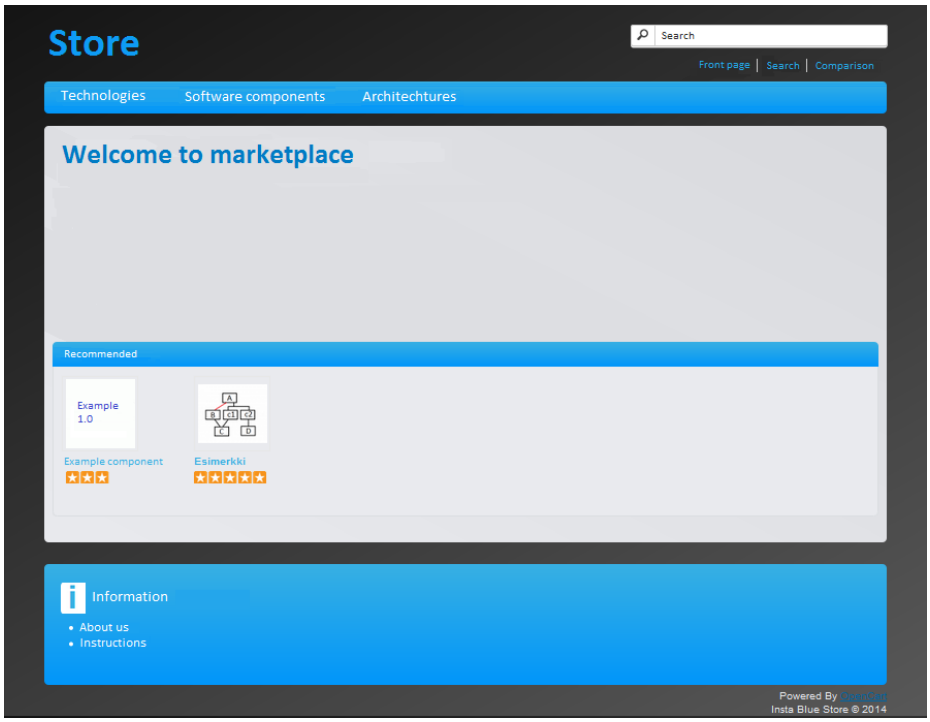

Fig. 3. Minimal user interface of the marketplace.

Within the system, software components are divided into different categories. The marketplace offers also an option to search for components. OpenCart's modified product pages are used for introducing components and they contain the essential information about the component deployment and usage. For instance, the description of a component might contain the following information:

- the version of the component,
- the purpose of the component,
- a contact person,
- a technical and functional description,
- a maintenance information,
- requirements and constraints,
- the location of the component, and
- licence term.

In addition, the OpenCart system supports user reviews, which in the marketplace are used for evaluating components.

6 Reuse process and marketplace

A company wide inner source approach was adopted for collecting and selecting the reusable components. The marketplace acts as one component in this process. The goal of the process is the systematic management of the reuse processes and the ability to control and guide the development of reusable components. This in turn aims to guide the selection of the technologies used. The process supports both the viewpoint of design-by-reuse as well as design-for-reuse [24] with the marketplace as a tool for advocating both views. The goal of the process is to identify and evaluate reusable components [25].

Figure 4 illustrates the complete reuse process within the organization with relation to the marketplace. The starting point of the component reuse is the company's internal developer community as they develop the product components. It is the job of the community, including the projects and product management, to initially recognize the reusable components and offer them to the Product Decision Board (PDB) for the evaluation process. As the company's business domain covers governmental organizations, some of the software components are tightly coupled with customer project-specifics and as such project-sensitive with a requirement of confidentiality. Naturally such components cannot be used in favor of other projects without major modifications.

The role of the PDB is to evaluate the components offered by the developer community and prepare a decision proposal in the form of Product Decision Card (PDC). PDB evaluates and selects the suitable components for the reuse and instructs the developer community to make the necessary changes to the component for reuse. After the decision on reusability the selected project will be responsible in modifying and documenting the component for reuse.

When the component modification for reuse is completed, it is added to the marketplace together with a description of the component. The description includes a list of features, an architecture description and user guide instructions. The administrator of the marketplace is responsible for making the component accessible for the company's developer community. That means adding the component and its artifacts to the marketplace. The customer projects can utilize the available components distributed through the marketplace. The developers can share reuse experiences with the chat tools offered there. The administrator ensures the reliability and correctness of the components. From the maintenance point of view the selected project takes care of the source code and the configuration of a component.

In the following, we evaluate PDB's reusability decision from three different perspectives – business, technology, and customers and stakeholders.

*B*usiness. From a business perspective the decision to productize is based on the evaluation of business model, product strategy, distribution strategy and

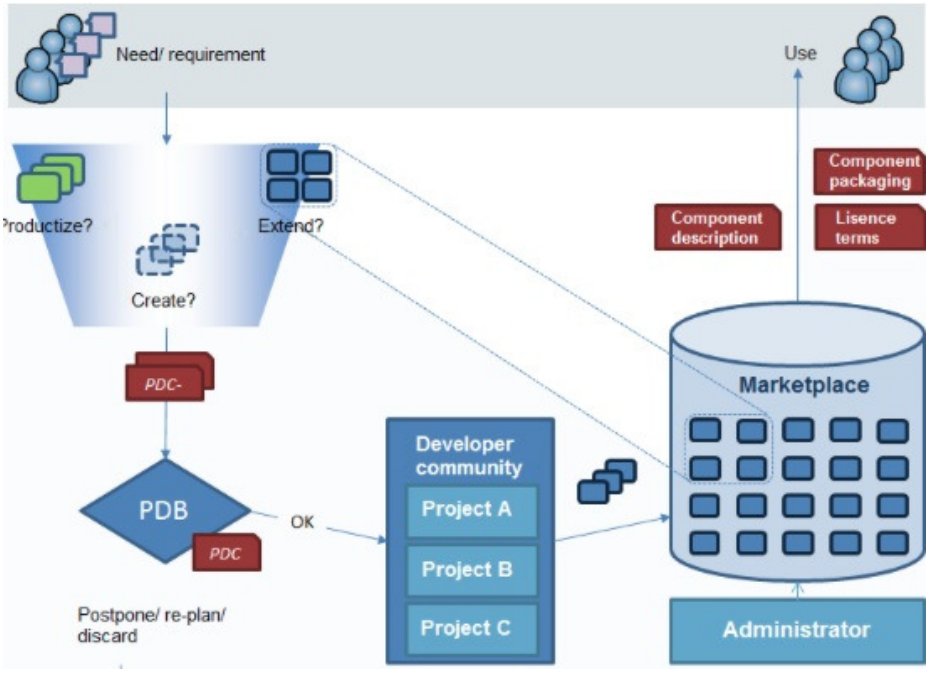

Fig. 4. Reuse process in the target organization

revenue model. In addition, PDB takes into account the productization costs required for changes and possible 3rd party license costs.

*T*echnology. From the technological point of view, the evaluation includes the technical feasibility and quality assessment of the code and the software architecture. The decision to productize a component also needs assessing the maturity and the component life cycle as packaging a component too early can cause the need for multiple updates which in turn will cause extra work and costs. On the other hand later updates can further lead to software incompatibility issues.

*C*ustomers and stakeholders. Overall, the customer projects and stakeholders' point of view the component must be utilized in a number of projects and component deployment process must be faster and cheaper than the creation of a new solution from the scratch.

7 Discussion

Next, we will revisit the four key processes proposed by Jacobson et al. and then we will discuss the marketplace in relation to them.

Create: The role of the process is to provide for the reusers. The marketplace aims to act as a clearer, easier and more accessible platform to share and

find reusable components. The development of the marketplace has been largely a part of the creation process. The needs of the developers were taken into account prior and during development. The requirements of the marketplace were prioritized based on developer needs.

Support: The role of the support process is to maintain the reusable components, the repository and the reuse processes. The marketplace plays a key role here. The role of the component descriptions is seen pivotal by the developers. Good, clear descriptions support reuse and make selection of components easier while the opposite can be quite harmful for the support process.

Manage: With the marketplace, the reuse process depicted in Section 5 provides the management for the other processes. The developer's see the role of the PDB and making correct reusability decidions as key.

Reuse: To what extent the marketplace increases reuse remains to be seen at this point. Based on the feedback, the developers see the marketplace as a welcome addition. The response of the developers will be discussed next.

Since the marketplace has just been deployed at the target organization, as a part of the prototype deployment an interview of the key stakeholders was conducted. Next, we discuss the feasibility of the marketplace through the estimation interview. In it three representatives of the intended target community – a software developer, a software architect and a project manager – was interviewed in an open interview session. The results indicated that there is a call for the marketplace at the organization. All interviewees seem to have identical opinions on the marketplace and how it should be developed further.

The content of the marketplace was seen as a key element in attracting the developer community. The content needs to be attractive to the developers and it needs to be maintained continuously. The interviews also highlighted that the use of the marketplace needs to be smooth in order to enable updating the component descriptions as well as to support creation of new components. In order to get content, all developers should be able to add components to the marketplace.

The active role of the developers was also valued. As many developers currently look for and reuse same components available online, a recommendation feature for them is seen beneficial. The feature would avoid a similar approach to be applied to the marketplace as it would relieve projects from doing the searching themselves in every case. Furthermore, an egalitarian approach to adding content to the marketplace was emphasized.

The reusable components themselves were seen as a valuable asset. The developers should however be able to estimate based on the description alone if utilizing the component is worth it or not. Hence the description should include the most restrictive knowledge that can affect the developers decision. Such are, for example, the licence, the price of the component and the process needed to reuse. Especially the ability the evaluate 3rd party components was seen as an asset, as it could save time and money. The fact that the components can include best practises and architectural decisions was valued.

8 Conclusions

At this point, the marketplace has been successfully deployed in the target organization. The developers comments on the marketplace and the organizational needs for it support the claim that it can make reuse more systematic and help to put in place and maintain the four key reuse processes. As future research we also wish that we have enough industrial data to validate these claims with experience data.

Presently, the marketplace is meant for company use only but as a future direction the possibility to share the marketplace with organization partners is considered. This lends way to future research on reuse over organization boundaries. The initial results are encouraging and the marketplace has shown its potential in enabling and supporting reuse over project silos to entire ecosystems that comprise of several companies. The next research steps are to collect data on the amount of reuse as well as evaluate what kind of reuse gets done; so far, the initial experiences look promising.

References

1. C. W. Krueger, "Software reuse," *ACM Computing Surveys (CSUR)*, vol. 24, no. 2, pp. 131–183, 1992.
2. D. Bauer, "A reusable parts center [technical forum]," *IBM Systems Journal*, vol. 32, no. 4, pp. 620–624, 1993.
3. W. C. Lim, "Effects of reuse on quality, productivity, and economics," *Software, IEEE*, vol. 11, no. 5, pp. 23–30, 1994.
4. J. Sametinger, *Software Engineering with Reusable Components*. Springer, 1997.
5. K. Pohl, G. Böckle, and F. V. D. Linden, "Software product line engineering," *Springer*, vol. 10, pp. 3–540, 2005.
6. J. Wesselius, "The bazaar inside the cathedral: Business models for internal markets," *Software, IEEE*, vol. 25, no. 3, pp. 60–66, 2008.
7. R. Goldman and R. P. Gabriel, *Innovation Happens Elsewhere: Open source as business strategy*. Morgan Kaufmann, 2005.
8. J. Dinkelacker, P. K. Garg, R. Miller, and D. Nelson, "Progressive open source," in *Proceedings of the 24th International Conference on Software Engineering*. ACM, 2002, pp. 177–184.
9. B. Boehm, "Managing software productivity and reuse," *Computer*, vol. 32, no. 9, pp. 111–113, 1999.
10. R. Prieto-Diaz and P. Freeman, "Classifying software for reusability," *Software, IEEE*, vol. 4, no. 1, pp. 6–16, Jan 1987.
11. W. B. Frakes and T. P. Pole, "An empirical study of representation methods for reusable software components," *Software Engineering, IEEE Transactions on*, vol. 20, no. 8, pp. 617–630, 1994.
12. D. Garlan, R. Allen, and J. Ockerbloom, "Architectural mismatch: Why reuse is so hard," *Software, IEEE*, vol. 12, no. 6, pp. 17–26, 1995.
13. I. Jacobson, M. Griss, and P. Jonsson, *Software Reuse: Architecture, process and organization for business success*. Addison-Wesley, 1997.
14. M. D. McIlroy, "Mass produced software components," in *Software Engineering: Report of a conference sponsored by the NATO Science Committee*. NATO, 1968, pp. 79–87.

15. W. B. Frakes and S. Isoda, "Success factors of systematic reuse," *Software, IEEE*, vol. 11, no. 5, pp. 14–19, 1994.
16. K. Schwaber, "Scrum development process," in *Business Object Design and Implementation*. Springer, 1997, pp. 117–134.
17. D. J. Anderson, *Agile Management for Software Engineering: Applying the theory of constraints for business results*. Prentice Hall Professional, 2003.
18. "Open source initiative," http://opensource.org/, last visited: September 2014.
19. E. Raymond, "The cathedral and the bazaar," *Knowledge, Technology & Policy*, vol. 12, no. 3, pp. 23–49, 1999.
20. D. Riehle, J. Ellenberger, T. Menahem, B. Mikhailovski, Y. Natchetoi, B. Naveh, and T. Odenwald, "Open collaboration within corporations using software forges," *Software, IEEE*, vol. 26, no. 2, pp. 52–58, 2009.
21. K.-J. Stol, M. A. Babar, P. Avgeriou, and B. Fitzgerald, "A comparative study of challenges in integrating open source software and inner source software," *Information and Software Technology*, vol. 53, no. 12, pp. 1319–1336, 2011.
22. K.-J. Stol, P. Avgeriou, M. A. Babar, Y. Lucas, and B. Fitzgerald, "Key factors for adopting inner source," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 23, no. 2, p. 18, 2014.
23. D. Ansorge, K. Bergner, B. Deifel, N. Hawlitzky, C. Maier, B. Paech, A. Rausch, M. Sihling, V. Thurner, and S. Vogel, "Managing componentware development – software reuse and the v-modell process," in *Advanced Information Systems Engineering*, ser. Lecture Notes in Computer Science, M. Jarke and A. Oberweis, Eds. Springer Berlin Heidelberg, 1999, vol. 1626, pp. 134–148.
24. S. Castano and V. D. Antonellis, "Reusing process specifications," in *Proceedings of the IFIP WG8. 1 Working Conference on Information System Development Process*. North-Holland Publishing Co., 1993, pp. 267–283.
25. A. B. Al-Badareen, M. H. Selamat, M. A. Jabar, J. Din, and S. Turaev, "Reusable software component life cycle," *International Journal of Computers*, vol. 5, no. 2, pp. 191–199, 2011.