Agreements for software reuse in corporations

Thijmen de Gooijer
Industrial Software Systems
ABB Corporate Research
Västerås, Sweden
thijmen.de-gooijer@se.abb.com

Heiko Koziolek
Industrial Software Systems
ABB Corporate Research
Ladenburg, Germany
heiko.koziolek@de.abb.com

ABSTRACT

Agreements for sharing of software between entities in a corporation have to be tailored to fit the situation. Such agreements are not legal documents and must address different issues than traditional software licenses. We have identified the topics such agreements should cover and in a case study found that the final content is heavily dependent on the structure and processes of the software organization. The presented work enables others to create guidelines for software sharing agreements tailored to their organization and shares lessons about the differences between software product lines and corporate software sharing and reuse.

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Reusable Software; K.5.1 [**Legal Aspects of Computing**]: Hardware/Software Protection—*Licensing*

Keywords

software reuse, software licensing, corporate agreements

1. INTRODUCTION

Code reuse and software product lines promise increased software quality and reduced development costs and time-to-market. Some companies achieved this holy grail of software engineering efficiency. For example, Hewlett Packard set up a software product line for the development of printer firmware and cut staff 75%, reduced the number of recorded bugs by 94% and reduced time-to-market by 50% [13].

Cost for software maintenance and development continue to increase, therefore we have looked at how to enable reuse of subsystems across business units. Subsystems cannot be copy-pasted like code snippets or binaries, but require integration effort. Reuse of subsystems across business units requires a different organization than a software product line (SPL). The SPL methodology is designed to create variations of the same product within one organization unit and requires thorough domain engineering and management of the variation points. We want to enable reuse across ABB business units that are responsible for different products and markets. Products can be software, contain software or be controlled by software, and often software is not the main component of the product. As a result there is a wild variety in size of development teams and their organization. Some software assets are developed by large teams, others by a handful of developers.

Business units that find opportunities for collaboration may need to setup license agreements when they reside in different legal entities. Hardware devices can be sold internally in a normal transaction, but for software there are concerns of access to source code, responsibility for maintenance, compatibility of future versions, and many more. Business units are not used to deal with such issues, because software development is not their core business. Since the agreements are not governed by law but by the corporation, using traditional licenses can undermine a sense of trust. Furthermore, the legalese in licenses is rarely appreciated.

We want to support reuse initiatives by making it as easy as possible to set up corporate agreements for software reuse. The agreements have fit any subsystem reuse case and have to be generic enough for ABB's diverse organization with around 145.000 employees in five divisions and dozens of business units. Business units often are geographically distributed organizations, which complicates sharing initiatives. Large, distributed organizations often face challenges with communication, collaboration, and trust that we also want to address [5]. To provide the needed support, we have created a guideline listing important agreement topics, e.g., integration fees, license grant, and maintenance support. For each topic the guideline lists several proven solutions which can be combined to draft a reuse agreement.

The first step we took to create our guideline was to review existing agreements and to collect success and failure stories from within and outside the company (Section 2). In the second step, we identified the important components of a sharing agreement and drafted alternative options for them as guidelines (Section 3). We refined our draft guidelines by mapping them to existing successful cases. This mapping is part of the guideline document and provides example solutions that are already successfully used and form a source of inspiration. The final step was to validate the guidelines by drafting a new agreement for an upcoming software sharing initiative (Section 4). At the end of this paper we share our lessons learned (Section 5), discuss related work (Section 6), and conclusions (Section 7).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

2. SUCCESS AND FAILURE FACTORS

We identify topics and suggested solutions for our guidelines based on experience from within ABB and that of the software engineering and software product line (SPL) community. In this section we describe the success and failure factors that we identified in existing ABB software sharing cases, but we first list success factors that we took from SPL literature (e.g., [4] [14]) and from software reuse literature (e.g., [8] [9] [12]).

- Plan reuse systematically instead of performing it opportunistically.
- Have management commitment and long term focus.
- Align SPL with business needs: an SPL must provide features that the market demands and can be sold.
- Select the appropriate domain for reuse: domains with high complexity and existing legacy promise the highest benefits.
- Tailor development processes and roles to SPLs.
- Agree upon a proper funding scheme agreed with all participating business units.
- Incorporate human factors and establish a proper organizational structure with clear responsibilities.
- Establish a proper infrastructure (e.g., good documentation, repositories, promotion) for SPL engineering.
- Ensure that the architecture is stable, well-documented and considers the common and the variable parts of all products in the SPL.

One example of a software sharing case inside ABB concerns a communication component and tool set, which are integrated by over a dozen business units. The software is provided as-is with source code and the agreement covers maintenance and support up to a certain maximum for a fixed yearly fee. Another example of software sharing is a structured sharing initiative in which business units jointly develop a component library. The participating business units share development and maintenance costs. One business unit acts as the lead and is in charge of administration. A potential issue is that the leading unit can exercise a stronger ownership and control over the software than the other participating business units.

From our survey of existing cases within ABB, we identified several success and failure factors. An important success factor is the implementation of standards. In several reuse cases, the reused software implements an industry standard or uses standards as guidance. The standard serves as a domain definition as used in SPL engineering.

Another success factor is 'as-is' delivery without shared development. For example, one unit ships documented source code to another without extensive support. This requires a close fit or a tight domain, but the provider does not have to make the software artifacts reusable and the impact of reorganizations on the consumer is minimized.

A third success factor is buying software components from external suppliers. Some business units outsourced development of software components that are not part of their core business and could not be bought of the shelf.

One re-occurring failure factor is the effect of internal reorganizations. Large corporations are always in flux through mergers and acquisitions, hiving off businesses, and frequent structural changes. It is challenging to sustain the required management support, processes, and knowledge throughout such changes. Ironically, reorganization can also create reuse across business units as development teams are split-up.

Another failure factor is lost sponsorship. This factor is strongly related to the first and can be a result of reorganization, but lost interest or turn-over among staff and management can have the same effect. The problem in these cases is that the software is integrated in several products and suddenly the common development stops, which leaves the software poorly maintained or implementations diverging and parts of the organization without sufficient knowledge of the source code.

The third and final failure factor is having an implicit product line. This happens for example when a reuse initiative or integration of products is started to generate a quick-win, but left unfinished without the organization in place. Because the organization is not anchored in the culture, the development unit is not made responsible for or empowered to build reusable components. As a result the consuming business unit has to spend considerable time and resources on integrating new releases of the software with its own product variant.

The identified success factors reinforce the importance of good business practice (e.g., make-buy decisions). All failure factors can be traced to failure to adhere to SPL best practices, providing informal evidence that SPL practices apply to more generic reuse situations as well.

3. SOFTWARE SHARING GUIDELINES

Based on the existing reuse cases and related work (e.g., [2]), we have identified five important topics for reuse agreements. An overview of these topics and examples of the sharing options that we have formulated is shown in Table 1. Before we discuss the topics and options, we define two terms. A (software) producer is the development unit sharing the software with a consumer. The producer is responsible for the development and/or maintenance of reusable software components. A (software) consumer is a unit that integrates a software component into their own software, i.e. reusing components created by a producer.

The first topic is license grant, which stipulates the rights the software producer grants to the consumer. Different levels of access can be granted to various artifacts such as binaries, source code, test cases, and documentation. Furthermore, conditions can be agreed upon for access to updates and for software asset escrow. One of the options we defined here grants the software consumer the right to binaries of the reused components and gives read-only access to the source code. The producer remains in control by reserving the right to change the source code and compile the software. Yet, the consumer has the possibility to debug its system by tracing errors through the code.

Payment is the second topic and concerns the remuneration the consumer pays to the producer in return for the granted reuse. While businesses take payments for goods or services delivered, it may be surprising to exchange money for software reuse within one corporation. However, business units may have their own budget. In this case, building reusable software does not deliver credit or value unless it is paid for by the consuming units. Payments between business units are regulated by internal and external guidelines, for example international tax codes. One possible model is to collect royalties on each sale of the consumer's product containing the producer's software and to have a fixed price

Topic	Example options
license grant	binary delivery with read-only source code source code delivery
payment	royalty payment / fixed maintenance fee share all development costs
support and maintenance	limited maintenance support no maintenance and support
ownership, administration, and confidentiality	company internal open source shared ownership
liability	full liability with the software integrator only indemnification for intellectual property violations

Table 1: Sharing agreement topics and example sharing options from our guideline

for basic maintenance of the software and to receive updates.

For the third topic the parties have to agree what level of support and maintenance is provided by the producer and what responsibilities the consumer has. This will, for example, enable the producer to plan resources to provide support. Furthermore, the producer's development organization is typically not used to deal with developer support requests from outside its own business unit. Therefore, communication channels and processes have to be agreed upon.

Consumer responsibilities may include the obligation to report any found bugs or a compulsory training given to developers that work on the reused software to reduce the demand for support by the producer's development team. The producer in turn commits to a certain rate of updates to the software and awards a certain priority to the consumer's change requests. An example option for this topic is to provide only limited support and maintenance meaning that the consumer receives major updates and can get support in case of bugs, but that no specific functionality will be implemented for the consumer.

The fourth topic is an aggregation, because the possible variation is limited. Legal ownership is always with the business unit's corporation, but there is the need to regulate the software development roadmap by clearly articulating who can influence strategic decisions. Processes for administration of agreements are stipulated, but how the agreement is enforced may be varied upon to some extent. Confidentiality levels are standardized within ABB, but have to be chosen for the various assets. In the example option of shared ownership, business units agree to share the responsibility for developing the software assets and the decision power over the roadmap for development. The share may be equal, but could very well be weighted by agreeing on a certain level of investment and associated voting power in decisions.

Finally, there is the topic of liability, which is defined as the extent to which the licensor has legally binding obligations, especially in the context of claims for intellectual property violations, damages, and injury. In a contract or agreement covering a product or service, the licensor typically attempts to limit its liability. It is common for commercial software license agreements to reject any liability. However, in case of software reuse within a corporation, the licensee (consumer) cannot drag the licensor (producer) to court. The question will have to be solved within the company and the structure, inter-dependencies, and culture of the business play an important role. Therefore, liability must be addressed in a software reuse agreement. One example option is to put all liability for claims about malfunctioning end-customer products with the reusing customer that sold the product containing the producer's software.

4. CASE STUDY

We have tested our software sharing guidelines by developing a new agreement and organization model for a growing software sharing initiative in ABB. Management of the initiative reviewed the proposal. The Software Sharing Initiative (SSI) is already successful on a small scale in developing components that conform to several industry standards. The SSI is currently gaining traction and more business units are joining. The growing organization thus needs a revised and clearer structure with an agreement to enforce it. Furthermore, several existing problems have to be tackled. For example, influence over the development is skewed towards the business unit that started the development, thus limiting the usefulness of the component to other units.

The new agreement for the SSI should extend the roles and responsibilities for the development to all participants. It should allow all business units to participate in the development and support a governance model that fits the increasing number of SSI participants. The agreement should also seek to include a funding model that is fair to existing participants and new entrants. Overall the agreement should take away any distrust between business units that normally do not collaborate. Finally, we were asked to explore the possibility of a company-internal open source model.

Based upon these requirements we drafted an agreement in which all participants in the SSI get full access to the software including its documentation, source code, and test cases. In return they commit developers to the SSI for a fixed duration and agree to share any addition, modification, or improvement to the software with the other participants. The payment is thus non-monetary. Participants also have the freedom to increase their development effort to speed-up development of features of particular interest to them.

A core development team reviews the quality of contributions by participants and ensures basic functionality is developed and maintained. The core development team also provides basic support for free and more extensive technical support for a fee. The proposed owner of the software assets is a corporate technology fund; while an SSI technology board decides on the development plans and takes care of administrative issues. All assets in the SSI have to be treated as confidential material and must not be shared with third parties outside ABB. Finally, liability is with the participant that sells the software components as part of its product.

5. LESSONS LEARNED

During the case study we learned that the guidelines had to be refined to match ABB's organization and that we should expand the proposal with processes and an organizational structure to achieve the desired properties and overcome current limitations. In the final SSI agreement proposal, we added flow charts to illustrate the processes, charts for the organization structure, and a RACI-matrix to document who is responsible, accountable, consulted, and/or simply informed for the different tasks in the reuse case. We supported all diagrams by explanatory text and rationale. The final proposal thus contains much more than an agreement, but this supports our view that an agreement has to be tailored for the culture and structure of an organization.

Some factors that we found to be important for internal sharing agreements, but do not show up or are interpreted differently in the SPL community. First of all, one should review the readiness of involved partners. Is their software development organization mature? Does their maturity match that of the partners? Are all willing to enter a software sharing initiative? Do the management structures connect?

The second factor is the role of software in the business units that share software. While many ABB products are increasingly software intensive, the role of software varies from being a supporting feature to being the core asset. Depending on the role that software plays in the organization, it is treated differently and with a different priority. It is important that the differences between sharing partners are made explicit to identify risks.

Third are differing requirements and especially quality requirements. For example, ABB develops products conforming to different safety integrity levels as defined in the industry standard IEC EN 61508. Thus, requirements on safe, fast, reliable, etc. must be clearly defined and agreed upon.

Finally, it is important to show willingness to align the development and release schedules and actually do so. This ensures that consumers use the latest version of the shared software, which reduces the number of versions that have to be maintained and supported concurrently.

6. RELATED WORK

Software license agreements are a well documented topic. Classen [3] and Chávez [2] both give an overview of the fundamental concepts, upon which we built our agreements. In the agreements we avoid legalese to make them easier to use.

D'Andrea and Gangadharan propose key topics for licensing web services and identify how web service licensing differs from software licensing [1]. Later they formalize the description of web service license options to describe them in machine interpretable form [7]. Our work does not focus on web services licensed to 3rd parties, but on systems software reused within a corporation.

Various work aims to help software license selection. Kaminski and Perry provide license patterns [10]. Their work differs from ours in that it aims to support the implementation of a license model in code. Ferrante gives an overview of licence models and trade-offs, but focusses on helping to be compliant [6]. Lindman et al. created a model to support decisions for an open source license taking into account factors such as business model and company size [11]. While our work does not account for all factors that they consider, we do not restrict our agreement options to open source.

7. CONCLUSIONS

Our experiences suggest that it is infeasible to define an agreement for internal software sharing that is as universal as open source licenses such as GNU's GPL. The type of agreement and the formality of the agreement that is required strongly depend on the level of trust between the business units and how much they already share. For example, some business units may share a division manager that can facilitate the process. In a more complicated example, business units may never have done business together before, come from different parts of the world, and hold strong prejudices against each other. The guidelines that we have created may empower software sharing that in turn should increase the productivity of our software development teams.

8. REFERENCES

[1] V. D. Andrea and G. R. Gangadharan. Licensing Web Services : The Rising. In *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, page 142, 2006.

[2] A. Chávez, C. Tornabene, and G. Wiederhold. Software Component Licensing: A Primer. *IEEE Software*, (October 1998):47–53, 1998.

[3] H. Classen. Fundamentals of software licensing. *Idea: The Journal of Law and Technology*, 37(1), 1996.

[4] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns.* Addison Wesley, Boston, MA, USA, 2002.

[5] J. Espinosa, S. A. Slaughter, R. Kraut, and J. D. Herbsleb. Team Knowledge and Coordination in Geographically Distributed Software Development. *Journal of Management Information Systems*, 24(1):135–169, 2007.

[6] D. Ferrante. Software Licensing Models: What's Out There? *IT Professional*, 8(December):24–29, 2006.

[7] G. R. Gangadharan and V. D. Andrea. Service Orientation : Licensing Perspectives. *Journal of International Commercial Law and Technology*, 4(1):1–11, 2009.

[8] R. Grinter. From local to global coordination: lessons from software reuse. In *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, pages 144–153, 2001.

[9] M. Griss. Software reuse: From library to factory. *IBM systems journal*, pages 548–566, 1993.

[10] H. Kaminski and M. Perry. The Pattern Language of Software Licensing. *SSRN Electronic Journal*, pages 1–41, 2005.

[11] J. Lindman, M. Rossi, and A. Puustell. Matching Open Source Software Licenses with Corresponding Business Models. *IEEE Software*, 28(4):31–35, July 2011.

[12] M. Morisio, M. Ezran, and C. Tully. Success and Failure Factors in Software Reuse. *IEEE Transactions on Software Engineering*, 28(4):340–357, 2002.

[13] L. M. Northrop, P. Clements, and E. al. SEI Framework for Software Product Line Practice. http://www.sei.cmu.edu/productlines/frame_report/

[14] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering. Foundations, Principles and Techniques.* Springer, Berlin / Heidelberg, 2005.