Designing the Inter-Organizational Software Engineering Cooperation: An Experience Report

Hans W. Nissen
RWTH Aachen, Informatik 5, Ahornstr. 55, 52056 Aachen, Germany
nissen@informatik.rwth-aachen.de

Abstract

This paper reports about experiences in managing the transformation from internal development and maintenance of software engineering tools towards an external one. We describe three different inter-organizational cooperation forms which differ in the distribution of development responsibilities between client and vendor – and which support the distributed design of three different classes of software products. An important finding was that even for software engineering tools which were extremely important for project success a carefully designed relationship model enables a successful distributed development.

1. Introduction and Background

This paper reports about experiences in establishing an inter-organizational cooperation between a world-leading telecommunication company and Indian IT service providers. The experiences we report focus on the selection of cooperation forms and the necessary organizational changes, but we are not looking at any financial aspect.

The subject matter of the cooperation was the huge set of proprietary software engineering tools used and developed within the telecom company. The cooperation goals from the customer perspective were to save money, to keep product quality and delivery precision, and to start a long-term cooperation with an external service provider.

Tools. The specific character of the software engineering tools influenced the selection of the form and the organizational set-up of the cooperation to a large extend. Therefore, we give in the following some background information on this subject matter.

Many products of the telecom company were based on a proprietary hardware platform and programming language. As a consequence, all supporting software engineering tools regarding that platform have been developed in-house during the last 20 years – all together the tool suite contained about 200 tools. Based on the companies strategic plans for software and system engineering, these tools were classified into three categories:

- *project critical tools which require continuous development and improvement activities*; examples are: compiler, simulated test environment, build environment. An erroneous or delayed tool delivery would cause major problems for the target projects developing telecommunication software.
- *tools which require further development activities*, but the new functionality is not project critical (i.e. the project could also survive without the development); examples are: version control system, modelling environment, traceability tools
- *tools which do not need further development activities but only maintenance activities* (i.e. bug fixing to some extend); examples are: editors, database applications to coordinate resource usage (e.g. error codes, signal names), document management system, fault tracking tools

Processes and Roles. The in-house communication and development processes were based on a formal client-vendor model: product managers search for (internal) project sponsors, collect requirements and develop the product release strategy. However, the internal tool development departments acted very flexible and accepted late and major changes to the original requirements. In case of faults and improvement suggestions, a fast and direct communication between tool users and tool developers was always possible. It was even seen as an advantage that this short feedback loop enabled high quality products. Especially for the project critical tools the direct and informal communication guaranteed the required quality and delivery precision.

2. Selection of Cooperation Form

The goal with setting-up an inter-organizational cooperation was to outsource a large part of the software engineering tool's development processes to external vendors under the constraint that neither the quality nor

the delivery precision must suffer.

The literature proposes cooperation models which differ in the distribution of development activities between client and vendor site. The outsourcing project analysed and discussed the following three cooperation models (cf. figure 1).

Classical contract model: The client delivers a set of specifications and the vendor implements or updates the software accordingly.

This model was selected to handle the non-critical tools without further development activities, i.e. tools which will go into maintenance mode. In this specific set-up the client does not deliver requirements for new functionality but requirements on fault corrections.

Implementation model: The client keeps much responsibility in-house: He specifies the requirements, identifies the impacts on the system components and produces the updated system and component specifications. The vendor is doing the detailed design on unit level and does the basic test. The integration test as well as the acceptance test is again performed by the client. This model leaves much intellectual work and responsibility with the client while the tasks of the vendor are limited more or less to the implementation part. This model requires good programming skills but less system management skills at vendor site.

This model was employed for the project critical tools because the control of these tools should remain within the company - at least the beginning of an inter-organizational cooperation.

Product management model: The vendor is taking over parts of the product management activities while the ownership stays with the client. The client participates in requirements specification and performs the acceptance test; the remaining parts of analysis, design, implementation and integration test activities are all done by the vendor. This model pushes most of the responsibility and work to the vendor and requires a very good understanding of the software product and the system environment at vendor site.

This model seemed to be most suitable for the non-critical tools with further development activities. The client company reduces the costs as much as possible and the vendor gets a chance to handle a product from beginning to end and to prove its competencies.

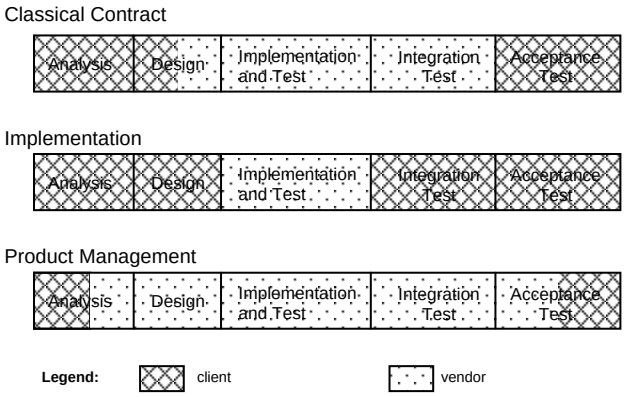

Figure 1. Three cooperation models

These different cooperation models require also different kinds of client-vendor relationships. The FORT framework described in [2,1] was identified to be valuable to characterise the resulting relationships. The FORT framework consists of two dimensions relevant for inter-organizational co-operations. The first dimension deals with the extent of ownership or control substitution by a vendor. The second dimension deals with the strategic impact of the portfolio. The four resulting types of cooperation relationships are support, alignment, reliance and alliance (cf. figure 2).

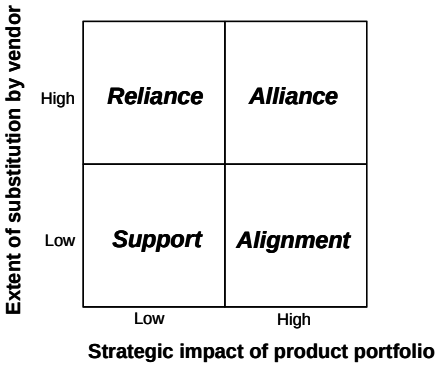

Figure 2. The FORT Framework [2,1]

In the support relationship, both the extent of substitution and the strategic impact of the outsourced products was low. The role of the external provider is therefore very limited. Within the alignment relationship, the substitution is low but the strategic impact of the outsourced products is high. This relationship is often used to obtain a service provider's specific expertise for a project. The reliance relationship highly involves the service provider in the client's processes. It therefore requires a high level of commitment from the vendor and a high level of reliance into the vendor competences from

the client. Finally, in the alliance relationship the vendor takes over a major part of the responsibilities for a product of high strategic value for the client. The two parties act as strategic partners with common goals [2,1].

The telecom company applied the product management model for the products with a medium strategic impact and included a high extent of substitution. This technical cooperation model is therefore best supported by the reliance kind of relationship. The implementation model on the contrast includes a low extent of substitution but is employed for products with a high strategic or risk impact. Therefore, this cooperation model requires an alignment kind of relationship. Finally, the classical contract model was designed for products with a low strategic impact and includes a medium extent of substitution. It can be located within the support relationship with a tendency towards the reliance relationship.

Summarising the selection of cooperation forms, the

- *project critical tools* were handled in an alignment relationship using the implementation model for distributing the responsibilities
- *non-critical tools with further development activities* were handled in a reliance relationship with the product management model
- *non-critical tools with no further development activities*, i.e. tools which go into maintenance mode, were handled in a support relationship with the classical contract model.

3. Changes in the Organization

The establishment of the cooperation with an external organization regarding the development of the software development tools led to some changes – in processes, roles and attitudes – of which we will mention only a few.

Processes and Attitudes. Within the requirements engineering process the developed specifications became more formal and the process itself was followed in a much stricter way. The habit of including late requirements into a development project – and sometimes changing the directions of the project by this completely – was abandoned. The product management process includes now as well the definition of formal acceptance test cases which did not exist before. These test cases and the requirements form an important part of the contract for product improvement; late changes would therefore cause unpleasant re-negotiations – in contrast to the flexible company-internal handling of such changes.

As new instances, monitoring processes have been installed. For the classical contract model, a simple variant in form of measurements on the delivered products has been chosen. For the two other cooperation models, a more sophisticated monitoring concept has been developed. Progress and product quality is measured during the whole development process, at clients and at vendor site.

Roles. The roles of product manager and system manager for the SW development products changed from operating only internal to acting as interface towards the vendor. From the vendors perspective the system manager become the most important role within the implementation model. He followed the implementation and answered the questions. At least in the first projects the goal was to exchange personnel in the way that ain the early phases employees of the vendor works together with the system manager at the client site. In later project phases the system manager will work partly at the vendor site to support and monitor the implementation activities.

The product manager is a key person for the product management model since he supervises the requirements acquisition and specification activities and takes the full responsibility from the client side.

Technical Environments. All the software development tools are integrated into a tool suite. This suite is still produced in-house whereby the integration test is the most important activity. This test verifies the programming interfaces, the external procedure calls, the side effects and the required documentation. The cooperation with external vendors required the external availability of the integration test suite such that the vendor himself is able to verify his tools. Within all cooperation models, faults reported by the users have o be forwarded to the maintenance department. The reporting and tracking environment used so far only in-house had to be available for external vendors, with the requirement that internal knowledge and information is really kept internal.

4. Conclusions and Lessons Learned

The establishment and the operation of the inter-organizational cooperation was successful regarding the stated goals, namely to save money by outsourcing the tool development and maintenance under the condition that neither the quality nor the delivery precision must suffer.

For the three different groups of tools three different cooperation models and relationship types have been implemented. The changes to some parts of the organization have been high regarding processes, roles and attitudes.

The lessons learned in this project can be summarised as follows:

- The applicable relationship types to the external vendor and their impacts to the organization must be carefully studied before a contract is signed or existing internal departments are discarded.

- There are major changes to the product management role and the requirements engineering processes and attitudes when moving from internal software development towards a global development model.
- An effective and honest change management is essential to successfully establish an inter-organizational cooperation which effects on employees.
- The establishment of an inter-organizational cooperation should be organised as a project with a set of clearly defined decision points, at least for the set of considered products, the designated cooperation forms, the resulting changes and their implementation strategy, the selected service provider. The management and the technical experts must be involved in all of these decision points.

References

[1] R. Kishore, H.R. Rao, K. Nam, S. Rajagopalan, and A. Chaudhury, “A Relationship Perspective on IT Outsourcing”, CACM, December 2003, Vol. 46, No. 12, pp. 87-92.

[2] K. Nam, , S. Rajagopalan, H.R. Rao, and A. Chaudhury, “A two-level Investigation of Information Systems Outsourcing”, CACM, July 1996, Vol. 39, No. 7, pp.36-44.

Acknowledgments

This experience report was funded in part by the German Ministry of Education and Research, BMBF, in project “VSEK” (FKZ 01 IS C65).