Optimizing Supplier Management in Global Software Engineering

Christof Ebert
Vector Consulting Services
christof.ebert@vector-consulting.de

Abstract
Global software engineering is the consequence of the rather friction-free economic principles of the entire software industry. Many companies start global software engineering (GSE) due to perceived cost differences. Most of these companies engage globally active outsourcing companies to achieve fastest ramp-up of their globalization targets. After a while into that business they realize that savings are much smaller and problems are more difficult to cure than before. Disillusioned many abandon their GSE activities. What has gotten wrong? GSE bears many challenges, specifically if done with an external supplier. We will look in this article to lessons learned from GSE to effectively and successfully manage supplier relationships. Specifically we use process maturity and CMMI as a guidance how to improve supplier relationships. Both experiences and practical guidance result from our work with clients in automotive and telecommunication industries. They can be readily implemented, specifically our checklists for effective supplier management.

Keywords: CMMI, GSE, distributed development, supplier management

1 Introduction
Global software engineering (GSE) is fast growing. It implies internal and outsourced development. Internal global software engineering happens within one legal entity, such as a company has application development in two countries contributing to the same product. Outsourced global software engineering means that one legal entity acquires services from another legal entity, such as a company has asked to another company to assist or entirely handle the application development. We will focus in this article on GSE activities with external suppliers. Many conclusions, however, can be also applied to internal GSE.

Labor cost varies across the globe. For the similar skills and output you pay in different places of the world a different amount of money per working hour or per person year. Looking to pure labor cost for comparable skills of educated IT-engineers, several Asian countries offer a rate of 10-40 % of what is paid for the same work time in Western Europe or USA. On top they offer such a huge amount of skilled and highly motivated engineers that it is impossible to not consider such potential for project planning. As a consequence the growth rates in some sectors are more than 20 % per year [1,2].

The 2006 ACM Job Migration Task Force report on globalization and software offshoring [3] underlines that globalization of the software industry will further increase driven by both information technology itself (e.g., skills and technology demands, market evolutions in emerging economies), by government actions (e.g., moving into IT sectors to reduce dependencies from raw materials such as in China) and by economic factors (e.g., labor cost differences).

With labor cost contributing to software development cost with two thirds and more, the savings should be in the range of 50 % [2]. That is at least what you can read in the newspapers. Reality has it that the savings are much lower. How can this paradox be explained? Why do so many externalized development activities fail? While the cost advantage and skill pool looks advantageous, global software engineering bears a set of risks that come on top to the regular project risks. Not knowing them and not mitigating them means that soon your project belongs to the growing share of failed global endeavors. This article will look into lessons learned from effectively managing supplier relationships in GSE-projects. It will discuss risks and provide many practical hints how to increase the success rate. Experiences are drawn from numerous such GSE initiatives in telecommunication and automotive industries. We have practically applied and evolved all mentioned guidelines and supplier management prac-
tices in working with clients in different industries. Due to the rather similar approach to GSE across industries with regard to selecting, managing and maintaining supplier agreement and processes, the mentioned results can be readily used in other companies.

2 Business Model and Challenges

The journey to global software engineering has begun but it is far from being clear what the stable end positions will be. Clearly some countries will come to saturation because global development essentially means that all countries and sites have their fair chance to become a player and compete on skills, labor cost, innovativeness and quality. Software engineering is based upon a friction-free economy with any labor being moved to the site (or engineering team) that is best suitable amongst a set of constraints. No customer is anymore in a position to judge that a piece of software from a specific site is better or worse compared to the same software being produced somewhere else in the world. In essence the old-economy labeling of “made in country x” has become a legacy thinking that does not relate to software industries. What counts are business impacts and performance, such as resource availability, productivity, innovativeness, quality of work performed, cost, flexibility, skills, and the like.

Fig. 1 shows the penetration of enterprises with different types of global development activities [2,3,4]. The horizontal axis provides the share of offshoring (as a proxy for the degree of GSE in an enterprise for an activity) and the vertical axis provides a view on the penetration of enterprises for a specific activity. The bubbles indicate activities that are typically subject to GSE.

For instance maintenance projects have already a penetration of more than half of all software activities worldwide (position on vertical axis) and it is typically done in an offshore environment rather than a single place in a highly paid country (right position on horizontal axis). Some activities clearly are not yet where they could be, such as new applications, or original equipment manufacturers’ (OEM) product development.

It is evident that the share of offshoring or globalization also depends on what software is being developed. While for mere IT applications or internet services the global development is fairly easy, embedded software still faces major challenges to distributed development. Only 30 % of all embedded software is developed in a global or distributed context, while the vast majority is co-located [3]. Not all software engineering tasks and projects benefit from GSE!

Big savings in GSE have been reported only from (business) processes which are well defined and performed already before offshoring started, and which need not much control [1,2]. This includes maintenance projects (under the condition that the legacy software has some type of description) where some or all parts could be distributed, technical documentation (i.e., creation, knowledge management, packaging, translation, distribution, maintenance) or validation activities. Development projects have showed good results in all cases where tasks have been well separated so that distributed teams would have direction and ownership.

Fig. 1: Usage and Impact of IT / Software Offshoring

Global development projects failed if tasks were broken down too much, such as asking a remote engineer to do the verification for software developed concurrently in another site [5,6,7,8]. Here distance effects and lack of direct communication slow down development rather than help it. The single biggest source of difficulties in GSE is related to communication across sites, bad communication hindering both coordination and insufficient management processes [9,10,11]. For instance, continuous integration of insufficiently veri-
fried and encapsulated software components fails if done remote to the parallel ongoing software development. Distributed teams working on exactly the same topic (e.g., the famous follow-the-sun pattern of developing a piece of software in different time zones) posed highest challenges for coordination and often resulted in severe overheads that would be measurable or tangible only later (e.g., features misinterpreted, insufficient quality, lack of ownership and responsibility, etc.).

With these challenges, reported cost reduction from GSE is much less than the above-mentioned potential of 50 % savings if the same process is split across the world with changing responsibilities [2,12]. Several such highly distributed software engineering projects reported a 10-15 % cost reduction after a 2-3 year learning curve. Initially outsourcing demands up to 20 % additional efforts [12]. Over 20 % of outsourcing contracts are cancelled in the first year [12].

To externalize insufficient engineering processes creates extra cost and learning curve driven delays – on both sides. These additional costs sum up to 20-40 % of regular costs of engineering. The learning curve for transferring an entire software package to a new team (e.g., location) takes 12 months [8,13,14]. Our own experiences [12,15,16] and research show that the effectiveness for software design and coding grows in a learning curve with 50 % effectiveness reached after 1-3 months and 80 % after 3-5 months. This obviously depends on process maturity and technology complexity. Each of the following bullets accounts for a 5-10 % increase to project cost:

- Supplier and contract management.
- Coordination and interface management, especially with fragmented / distributed processes.
- Project management and progress control.
- Training, knowledge management, communication.
- IT infrastructure, global tools licenses.
- Liability coverage, legal support.

### 3 Supplier Selection for GSE

Working with an external supplier that provides the engineering services in an offshore (or nearshore) scenario is shown in Fig. 2. The supplier of the (global) engineering services will from day one build strong interfaces to the major functions in the product life-cycle of its client. Again, these interfaces are shown with arrows. Interface management is the clear professional need of the supplier, while for the client it looks certainly like overhead. And frankly it is overhead, but born out of risk mitigation of the supplier who otherwise would fear that changes would continuously ripple from the client to his organization making it impossible to keep service level agreements (SLA) and delivery commitments. Only those interfaces cost an additional 10-20 % (depending on the maturity of both client and supplier) on top of regular project cost without any value-add as seen from the client. With several other suppliers involved for component deliveries such overheads can grow into the 30 % range. Needless to say, people management and competence management are handled inside the supplier (not shown for the external component supplier which is even more separated) on the basis of forecasts delivered by means of the contract and regular client stakeholder reviews.

![Fig. 2: Global software engineering with external supplier (outsourcing)](image)

When selecting external suppliers, a key prerequisite is in identifying which supplier fits best to your needs. There are a few simple rules to follow:

- Select a supplier that fits with its size and business model to your own company. For instance, a very big business process outsourcing supplier might be less interested in providing specialized services to a small company. The supplier might be willing to do so, but after a while behaviors will be less supportive to the needs of the small or medium enterprise. They would have done better with an intermediate used to working with small enterprises or rather taking a supplier specialized in this type of management.

- Select a supplier with sufficient process and methodology know-how. As a rule the supplier of engineering services must have high process maturity. Demand a recent CMMI appraisal valid for the entire company and evaluate results (we will come back to the CMMI in the next section of this article). At least it should indicate a maturity level 3 because this is a solid foundation of effective engineering and project management...
processes along the product life-cycle. Preferably it should indicate a maturity level 5 that many of the external suppliers today have achieved.

- Assure process flexibility. It is of not much help if you select a supplier that demands to use exactly the same processes and tools across all projects. A good supplier is capable to adjust its own processes and interfaces to your tools. Of course they should be able to help you in improving processes and optimizing tools, but this is decided on a needs basis.
- Select a supplier with sufficient domain knowledge in your own field. Having domain expertise allows putting skilled engineers to your project that need less learning to understand technical aspects related to the product or service rather than only to the design and programming language.
- Demand a list of engineers working on your project with skills, current subject experience, previous projects, etc. Insist that these engineers are allocated to the project in case you want to build skills over a time. Note though that defined engineers and names typically increase cost per head because it reduces flexibility at the supplier side.
- Use a supplier that is physically present at your own site. It is often substantial to continuously travel to the supplier site or having only videoconferences. Having not only the supplier’s sales team locally but also some engineering skills eases requirements and change management.

4 Optimizing the GSE Life-Cycle

GSE needs to consider the entire product lifecycle. Fig. 3 shows the relevant phases along the product life-cycle and the respective activities related to supplier management. Four major phases are distinguished, namely supplier strategy, supplier selection and contracting, contract management, and evaluation and relationship management.

A client or customer organization must provide a realistic and precise expectation of functional and non-functional requirements (e.g. reliability). They should clearly state that payment will be provided only for systems that meet the agreed upon functionality (e.g., requirements, acceptance tests, SLA conditions). And they should demand milestone presentations of progress for continued funding.

Supplier organizations on the other hand must insist on a signed contract with requirements. They must agree before contract signature on clear and reasonable acceptance criteria. The contract must be explicit that the supplier owns the software until final payment. They must clearly agree on liabilities and support after handover. They have to express disagreement and unrealistic conditions openly and not continue with diverging assumptions. They should always strive for win-win results and therefore offer compromise approaches, once needs are understood. In cases of component delivery they should include a software key that will operate after the date of contracted software acceptance.

Fig. 3: Supplier agreement management

As a client you should always consider the golden rule of supplier management: You pay for what you get. Don’t get trapped into contracts that look “cheap” and later bring tons of extra cost due to lousy processes and insufficient delivery quality. Preconditions of any successful supplier management are good processes on both sides, i.e., for the client and the supplier. Insufficient client processes cannot be externalized. They will not scale up from a single site to several sites. Often those low-maturity processes can be handled in localized development without many overheads due to colocated teams, but will fail with globalization.

In a recent survey with automotive suppliers and OEMs (i.e., those who engage the suppliers) with companies such as DaimlerChrysler, Ford, GM, Bosch,
Delphi, Denso, Johnson Controls, Lear, SiemensVDO, TRW, Valeo and Visteon, we found that patterns very much resemble across GSE users [17]. And they are the same like in other software and systems industries. Fig. 4 shows these dependencies.

Of the participating suppliers a vast majority have chosen CMMI to improve their software engineering and management processes. The Software Engineering Institute’s (SEI) Capability Maturity Model Integration (CMMI) provides a framework for process improvement and supplier capability assessments [18]. It defines five maturity levels together with a process improvement framework to be used for own improvement programs and to evaluate suppliers.

Companies have started to realize that change is critical: If you stand still, you fall behind! The business climate and the software marketplace have changed in favor of end users and customers. Companies must fight for new business, and customers expect process excellence. An industry-proven framework like the CMMI offers the benefit of improvement without getting lost in theory. CMMI offers concrete guidance in the form of structured goals, plus a wealth of best practices to implement. It presents maturity and capability levels to set up a goal-driven roadmap and a well-defined appraisal technique to assess processes and benchmark. As a whole, the industry is becoming more mature. It’s all about business: Your competitors are at the same place as you (if not ahead). The goal is to further improve planning and decision making, lower costs, adhere to schedule, and improve product quality.

Industries and companies of all sizes worldwide use CMMI, so it provides a huge body of knowledge on topics along the entire product life-cycle. Based on broad feedback of this worldwide user community the SEI recently updated this content to CMMI v1.2 [18]. Being originally based on the Software-CMM which appeared twenty years ago, it’s today also governed by the International Organization for Standardization (ISO) to assure that suppliers are assessed comparably on a global basis.

CMMI levels of achievement among the Tier One suppliers are typically higher than those of the interviewed vehicle manufacturers with most reporting maturity levels of 2 by the end of 2006 and goals of maturity level 3 by the end of 2007. The maturity difference between OEMs and suppliers causes inefficiency due to mismatch at their interfaces and the risk to lose control for the less mature partner (see Fig. 4). Thus OEMs have to speed up their process improvement in order to keep up with their suppliers.

For your own processes – both in supplier management as well as own engineering processes along the product life-cycle – and their improvement, we strongly recommend using the CMMI. Contents of the CMMI range from selection and contract management to building a shared vision and effective collaborative teams. Why do we recommend using the CMMI? The CMMI has rich evidence from big global systems development projects with different contractors working on one assignment. Many companies have been using the CMMI over the past years in both systems, hardware and software engineering, and they all found strong evidence that improves engineering performance – both seen from a supplier or from a client perspective.

Companies that do not meet the CMMI have the following consequences:

- **Process Maturity Supplier**
  - Low
  - High

- **Overheads**
  - Lack of downstream integration, rework cycles

- **Win-Win**
  - Process integration, mutual optimization

- **Failure**
  - Dysfunctional interfaces, frictions, overruns

- **Replacement**
  - Low supplier performance, selection of better supplier

**Fig. 4: Maturity mismatch between OEMs and suppliers causes inefficiency**

As a company embarking on GSE with external suppliers, assure that your processes are in good shape before they are used in a global context! Rigorously enforce using the agreed standard process that relates to a CMMI Maturity Level 3 organization pattern. Provide an interactive process model based on accepted best practices that allows tailoring processes for the specific needs of a project or even a team.

Organizations on CMMI maturity level 1 or 2 should not expect that global software engineering would yield much benefit. Instead it will reveal major deficiencies in processes and workflow, which create all type of difficulties, such as insufficient quality, delays, additional cost, cancelled offshoring contracts, demotivated workforce in both places (previous and new), and many more. The only viable alternative for such low-maturity organizations is to work closely with a consultant and ramp-up the own processes before proceeding with GSE. On the other hand we have seen from previous ramping-up big internal software teams in Eastern Europe, India and China that solid processes
not only accelerate introduction of GSE but also serve as a safety net to assure right training, good management practices, etc. We observed one controlled experiment in a global telecommunications OEM when ramping up their offshore development teams. The building of such globally distributed development team was fastest and most reliable in the case where the demanding organization was on maturity level 3. The same was done with lower-maturity demanding organizations with the effect that the maturity level 2 organization could manage with some external support, while the maturity 1 organization failed due to highly inefficient interface frictions and lots of rework.

The very observation that higher maturity is a gain for both client and supplier in a GSE program is the driver behind many ongoing improvement programs, such as at General Motors. GM’s electronics VP recently pointed out that all GM must be at CMMI maturity level 3. But that is not the whole story. In order to be more effective in their collaboration with external suppliers, GM expects that everybody involved from outside must work along these same lines and have a continuous improvement program up and running [21].

The exchange of information between supplier and client must be carefully planned. A change management tool is not enough because engineering demands collaboration on content and knowledge. Plain supplier management platforms as they are offered today for handling online market places and tenders are also insufficient due to their limitations in sharing engineering information. You will need rules and workflow support for documentation, design reviews, change management boards, etc. We recommend to install workflow management and online accessible project, work product and process information to assure proper knowledge management. Interactive process models, such as RUP and others have proven very helpful to communicate and install processes [19,20].

The following checklists provide some sample checklists for supplier and contract monitoring.

Our first checklist is used to highlight specific supplier-created risks during contract execution. They are a kind of formalized guts feeling to smell changes which (specifically if several appear) point on growing risks.

- Sudden behavioral changes
- Contractual agreements are not kept
- Difficulties and issues are not communicated
- Frequent rejection of inputs, specifications, etc.
- Above average turn-over rate of engineers on your projects
- Reduced contact with supplier senior management
- Demand to re-prioritize requirements
- Overly exact and restrictive interpretation of SLA
- Increasing amount of escalation
- Financial situation of supplier worsens
- Other clients leave your supplier
- Supplier gains new and more relevant clients

The second checklist provides some checks for supplier evaluation during an ongoing project.

- Is progress according to agreed milestones and deliverables?
- Are right skills and engineers available as agreed?
- Is technical expertise on right level?
- Are agreed quality levels of deliverables proven?
- Are the budgeted cost and schedule kept?
- Is quality, cost and content of work products adequate?
- Which risks materialize? Which risks are mitigated?
- Are agreed standards and processes implemented?
- Is security and intellectual property sufficiently protected?
- Are governance mechanisms installed and followed?
- Which improvements are proposed by supplier?
- Is there any way to improve relationship management?

The third checklist finally is used at contract termination or project end to summarize lessons learned for future supplier management activities and process improvement.

- Did the supplier give the perception of being sufficiently qualified?
- Have schedule and budget constraints been kept?
- Have all deliverables been according to SLA and quality levels?
- Has effort been in line with estimates? Why not?
- Which risks materialized? Which risks could be mitigated?
- Which improvements are proposed by the supplier?
- Which improvement is suggested by own team?
- Has the work split and allocation been adequate?
- Are there possibilities to improve relationship management?
- Are there possibilities to improve communication?
- Is this the right supplier to grow with or to continue with?

Effective supplier management is very much depending on the oversight and governance capabilities of an organization. Too often suppliers are believed to deliver according to specifications and SLA, and suddenly the client realizes that this was mere wishful thinking. Here some concrete measurement-oriented hints. They are written from a client perspective but of
course should be considered when working as a supplier – because this is what you clients will demand.

- Set up clear and measurable service level agreements. Ensure that this SLA contains all that matters for you in the contract.
- Insist on periodic reporting according to the SLA.
- From the beginning define thresholds that establish when and how insufficient performance will be escalated.
- Measure supplier capability or demand such measurement based on industry standards, such as CMMI.
- As a supplier or customer move towards high maturity product development by using the CMMI and its maturity level 3-5 concepts of process excellence and quantitative management. Do not stagnate on maturity levels 1-3 as you will be eventually replaced.
- Relate value you receive from suppliers to the risk and cost of the delivered services or components. Manage the risk of lock-in and dependencies that could create extra risk and cost.
- Implement contract evaluation after each single project. Go beyond the qualitative checklist and report into measurements and fact-based lessons-learned.

Consider sufficient time and budget (resources) for training the supplier on your processes. A very strong training tool is the Scrum process with short team meetings every day where recent results and next steps are briefly reviewed. Any uncertainty should be brought up in such reviews, which should take not more than 15-30 minutes and can be conducted even per telephone conference across sites. Build a supplier program management to handle the necessary review and decision processes. Agree with your supplier review and acceptance processes to assure the right quality level. Installing such processes after the contract signature will create the perception of policing the supplier. You can ask third parties in case of questions or needs for escalation.

Globally distributed software development is highly impacted by work organization and effective work split. Working in a globally distributed project means overheads for planning and managing people. It means language and cultural barriers [10]. It creates jealousy between the more expensive engineers being afraid of losing their jobs, while forced to train their much cheaper counterparts. The barriers to such harmonization and cooperation are not to be underestimated. They range from language barriers to time zone barriers to incompatible technology infrastructures to heterogeneous product line cultures and not-invented-here syndromes. An obvious barrier is the individual profit and loss responsibility that in tough times means primarily focusing on current quarter results and not investing in future infrastructures. Incumbents perceive providing visibility a risk, because they become accountable and more subject to internal competition.

Although there are no patent recipes for GSE and work allocation, many experiences from previous projects indicate what we might call "typical configurations". Such configurations are shown in Fig. 5.

The first column to the left indicates the "operational scenario" of global product development and operations. It starts with the beginning of the product (solution) life-cycle and moves to installation and operation towards the bottom of the table. The second column shows the most appropriate business model for such an operational scenario. The next column indicates how external suppliers might be included. Obviously, external suppliers do not fit in all scenarios, depending on intellectual property and dependencies exposure, but also related towards risk management of future growth. The learning curve duration and the breakeven period depend on these scenarios and are summarized in the subsequent columns. The last column finally portrays how many parties (external or internal) are most appropriate. Needless to say, most scenarios are most effectively handled with a small number of contributors – except such cases where the contribution can be well isolated and decoupled from overall project flow and risks (e.g., software components or platforms which are selected and evolve in parallel but without critical dependencies).

5 Conclusions

Managing global software development is not easy and has high-risk exposure to lowering overall productivity. Still the positive impacts should not be forgotten. A major positive effect that is boosted by going global is innovation. Engineers with all types of cultural backgrounds actively participate to continuously improve the product, to innovate new products, and to make processes more effective. Even with the slightly more complex decision making process behind, achievements are substantial if engineers of entirely different education and culture try to solve problems. Best practices can be shared, and sometimes small changes within the global development community can have big positive effects.

This article has investigated how supplier management impacts GSE and what lessons can be drawn to improve supplier management in GSE scenarios. GSE makes deficiencies more visible and it amplifies weaknesses. These deficiencies are always there with or
without GSE, but in a global and distributed context they have more impact.

We have shown evidence that effective process management within a company improves its capability of doing GSE, both being a supplier of engineering services as well as being the client using such services. Specifically we introduced the CMMI being a model that provides objectives to be fulfilled by engineering and management processes in order to be successful in a global software engineering landscape. Using the CMMI effectively means to select from the model what matters to your business needs (e.g., being a small or large supplier, being a client or supplier, being an integrator or a component supplier, delivering software or services, etc.). CMMI training courses and experience of CMMI professionals – be it engineering process group leaders or consultants – will help you to stay on track. Always remember that successful change needs not only a technical framework for guidance (that is, this book) but also concrete improvement objectives and – above all – leadership to drive the change in the direction you need.

As a supplier you should strive for high maturity, for several reasons, primarily:

- The market attractiveness in software-driven industries is extremely high, due to low entry barriers and continuous push for innovative products. You are in strong global competition for excellence.
- Suppliers have recognized that better process maturity ensures better schedule and SLA performance, productivity and quality. Why should not they demand it along their supply chain.
- High market attractiveness continues to push to cost reduction and efficiency improvement. Many new entries start each day with similar business ideas that drive your own company. If you do not continuously improve both products and processes, they will do it. The best example is the move of high-technology products to Asia, as there is a much higher competitive pressure for high process maturity than in North America or Europe.

Needs for global engineering must be carefully balanced with additional cost that might occur only at a later point. This includes staff turnover rates which vary heavily across the globe; cost overheads related to traveling, relocation, communication or middle management or redundant development and test equipment; unavailability of dedicated tools that allow for globally distributed tools and work environments; impacts of the learning curve that slows down with more locations involved; cultural differences which can impact work climate; insufficient language skills; different legal constraints related to work-time, organization, or participation of unions; and building up redundant skills and resource buffers to be prepared for co-located teams and for unforeseen maintenance activities.

History has shown us time and again that mixing genes is the best thing that can be done in the path of evolution. Or in the words of Charles Darwin, who was one of the first truly globally acting scientists: "It is not the strongest of the species that survive, nor the most intelligent, but the ones most responsive to change." Globalization is about the same concept…

6 References

This report results are confidential and can be ordered by members via www.rtec.executiveboard.com


<table>
<thead>
<tr>
<th>Task</th>
<th><strong>GSE (business) model</strong></th>
<th><strong>Supplier model</strong></th>
<th><strong>Learning curve</strong></th>
<th><strong>Breakeven period</strong></th>
<th><strong>Number of partners / sites</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition and analysis of new business models</td>
<td>Preferably onshore; should be co-located</td>
<td>Consultant; preferably own organization</td>
<td>Long</td>
<td>Long</td>
<td>Few</td>
</tr>
<tr>
<td>Product definition; platforms and applications for resale</td>
<td>Onshore, close collaboration</td>
<td>Consultant; preferably own organization</td>
<td>Long</td>
<td>Long</td>
<td>Few</td>
</tr>
<tr>
<td>Development of internal (ICT) applications</td>
<td>Offshore</td>
<td>Typically outsourcing</td>
<td>Short</td>
<td>Middle</td>
<td>Few-many</td>
</tr>
<tr>
<td>Product development (generic)</td>
<td>On- / near- / offshore</td>
<td>Typically outsourcing or own dvmt center</td>
<td>Middle</td>
<td>Middle -long</td>
<td>Few</td>
</tr>
<tr>
<td>Product development (embedded; complex)</td>
<td>On- / near- / offshore; single project should be co-located</td>
<td>Typically own development centers</td>
<td>Short</td>
<td>Middle -long</td>
<td>Few</td>
</tr>
<tr>
<td>Validation of software</td>
<td>On- / near- / offshore; tasks test and development should be co-located</td>
<td>Typically outsourcing or own test center</td>
<td>Middle</td>
<td>Middle</td>
<td>Few</td>
</tr>
<tr>
<td>Maintenance of internal applications</td>
<td>Offshore</td>
<td>Typically outsourcing or own dvmt center</td>
<td>Middle</td>
<td>Middle -long</td>
<td>Many</td>
</tr>
<tr>
<td>Maintenance of products</td>
<td>Near- / offshore</td>
<td>Typically outsourcing or own dvmt center</td>
<td>Middle</td>
<td>Long</td>
<td>Few</td>
</tr>
<tr>
<td>Selection and installation of software and infrastructure</td>
<td>On- / nearshore, close collaboration</td>
<td>Consultant; preferably own organization</td>
<td>Short</td>
<td>Short-middle</td>
<td>Few</td>
</tr>
<tr>
<td>Operation of infrastructure</td>
<td>On- / nearshore</td>
<td>Typically outsourcing or own IT center</td>
<td>Short</td>
<td>Short-middle</td>
<td>Few</td>
</tr>
<tr>
<td>Operation of internal applications</td>
<td>On- / near- / offshore</td>
<td>Typically outsourcing or own IT center</td>
<td>Short</td>
<td>Middle</td>
<td>Few</td>
</tr>
</tbody>
</table>

**Fig. 5:** GSE work allocation and typical configurations