Assessments in Global Software Development: A Tailorable Framework for Industrial Projects

Frank Salger
Capgemini sd&m
Carl Wery Straße 42
81739 Munich, Germany
++49/(0)89 63812-221
frank.salger@capgemini-sdm.com

Gregor Engels
University of Paderborn
Warburger Str. 100
33098 Paderborn
+49 – 5251- 60 3336
engels@upb.de

Alexander Hofmann
Capgemini sd&m
Carl Wery Straße
81739 Munich, Germany
++49/(0)89 63812-310
alexander.hofmann@capgemini-sdm.com

ABSTRACT

Assessments are an effective technique for software quality assurance. As global software development (GSD) becomes the standard, an assessment framework must be flexible to support different sourcing and shoring models. Although much work exists on inspections and reviews, an assessment framework which addresses these challenges is missing. We present a systematic yet flexible assessment framework. The paper contributes: i) The description of our assessment framework which addresses four challenges: *Appropriateness* of a software requirements specification (SRS), *viability* of software architectures and SRS, *wholeness* of work packages, and *compliance* of results with predefined quality objectives. ii) A detailed explanation how the assessment framework can be tailored to support offshore and outsourcing scenarios. This paper describes the result of a two years research initiative at Capgemini sd&m and serves the practitioner to implement assessment frameworks according to his needs. We also discuss open research questions of high relevance for the software industry.

Categories and Subject Descriptors

D.2.9 [**Management**]: Software Quality Assurance

General Terms

Management, Measurement, Verification.

Keywords

Global Software Development, Assessment

1. INTRODUCTION

The smooth transition from requirements down to design is always crucial for project success. Getting software requirements specifications (SRS) right is one mission critical activity [1, 2].

Defining an architecture which satisfies the architecturally significant requirements and the tradeoffs between them is another important task [3]. Hence, it must be carefully evaluated, how a project carries out these two steps. Reviews and inspections have been proven to be one of the most effective and efficient techniques to evaluate both, SRS and software architectures [4].

On the other hand side, errors are often introduced at organizational interfaces, like e.g. between successive process steps and at synchronization points where work products and responsibilities thereof are handed over between teams. Research has shown that in global software development (GSD), the careful definition of work packages and the management of handovers thereof become especially important [5-7].

This importance is further increased in modern software development where offshoring and outsourcing steadily increases [8] and becomes the norm rather than the exception [9]. Basically, offshore development means to transfer work packages to a different country, which does not necessarily span different companies. On the other hand side, outsourcing spans different companies but does not necessarily involve distributing work over different countries.

Extensive shoring and sourcing makes it also necessary to align terminology between the involved parties. This is especially true for work products like for example a software requirements specifications, as they constitute a central artifact over which most of the communication between the different stakeholders involved in GSD is channeled [10-12].

Summarizing the situation, we can identify four critical challenges for a smooth transition from requirements to design in GSD:

- *Appropriateness* of the SRS concerning structure and terminology from the point of view of all relevant stakeholders involved in the project.
- *Viability* of the SRS and the software architecture concerning all downstream activities.
- *Wholeness* of the work packages from the developer point of view.
- *Compliance* of the development results to defined quality objectives and acceptance criteria.

These four challenges have been observed to be of critical importance to overall project success. The relevance of *appropriateness* of a SRS has been discussed in [13]. Many works investigate how to assess the *viability* of a SRS and a software architecture (see, for example, [3]). *Wholeness* of work packages has been addressed in [14], and the importance of rigorously checking *compliance* of development results in GSD has been affirmed in [6, 7, 11, 13].

However, a comprehensive assessment framework which addresses all of the four requirements is missing. Moreover, different project settings demand different assessment rigor. For example, in offshore outsourcing projects, a higher assessment rigor will be necessary than in pure intra-organisational projects.

Our approach offers the following two main contributions:

- **Assessment-Framework:** Three assessment types are bundled into a comprehensive framework to assure the flow down of requirements engineering to design.
- **Tailoring rules:** The framework is backed with a pragmatic and easy to follow categorization on how to adapt the framework to different offshore and/or outsourcing project settings.

This work summarizes the outcome of a two years research initiative carried out at Capgemini sd&m, which involved numerous interviews with senior architects, dig outs of completed GSD projects, as well as literature research and empirical research conducted at the Capgemini Group [15].

The paper is organized as follows. In the next section, we describe the company and the kind of systems our assessment framework is geared to. We then present an overview on the framework and the overall goals we pursue with it. Sections three to five explain the constituent assessment types in detail: The 'Specification Check' (section 3), the Quality Gates (section 4) and the 'Handover Checkpoints' (section 5). For each assessment type we discuss related work and state our contribution to the existing body of research. In section 6, we show how the framework can be tailored to different sourcing and shoring scenarios. We draw our conclusions in section 7 and conclude with discussing remaining open problems with high relevance to global software development in the industry.

2. OVERVIEW

In order to be valuable for other practitioners and researches, we first explain the context of this work.

Company:

Capgemini sd&m is the Technology Services unit of the Capgemini Group in Germany, Switzerland and Austria and offers its customers end-to-end process and software solutions that have a crucial impact on their competitiveness.

At the moment, Capgemini sd&m employs about 2000 software engineers. Capgemini sd&m is part of the Capgemini Group which is present in over 30 countries and runs offshore development centres in numerous countries. Two 'dedicated teams' (at Wrozlaw, Poland and at Mumbai, India) exclusively work for projects at Capgemini sd&m. In cases, Capgemini sd&m also outsources development work to onshore third party suppliers. Such different project settings – which involve both, offshoring and outsourcing of engineering work –where one reason to develop the tailorable assessment framework presented in this paper.

System type:

The assessment framework presented in this paper is geared towards custom software development projects of large scale business information systems (BIS). Such BIS can encompass up to several hundred use cases, user interfaces and domain objects, as well as complex concepts for print output, batch processing, archiving, authorization and multi-tenancy.

The BIS we develop at Capgemini sd&m support critical business processes of our customers but are usually not safety-critical (e.g., critical to life). As mentioned, the simple yet useful idea behind this framework is that it addresses the four critical challenges on the way from requirements down to design: *Appropriateness*, *viability*, and *wholeness/compliance*. The framework leverages three 'assessment types' to address these challenges:

1) The first assessment type, called the 'specification check' (SC), assures the *appropriateness* of the SRS, especially from the offshore development team point of view. The SC is applied at the stage, where the information model and the overall structure of the SRS are defined.
2) The second assessment type is needed to assure the *viability* of core work products. Here, we use our 'quality gates' to carry out in-depth inspections of SRS and software architectures [16, 17].
3) The third assessment type, the so called 'handover checkpoints' encompasses two handover checkpoints and addresses the issues of *wholeness* of work packages and *compliance* of development results to defined quality objectives and acceptance criteria.

Table 1 shows a high-level view on the three different assessment types, differentiating them along the dimensions 'scope' and 'goal'.

Table 1. High-Level View on Assessments

Assessment	Scope	Goal
Specification Check	SRS	Assure *appropriateness* of SRS structure for all stakeholders, in particular offshore development team. Ensure that common understanding of core concepts has been achieved.
Quality Gates	SRS/ Architecture	Assure *viability* of core work products by in-depth assessment of various quality aspects and their tradeoffs.
Handover Checkpoints	SRS/ Architecture / Plans, schedules, metrics	Assess *wholeness* of work packages and definition of reasonable acceptance criteria. Assess *compliance* of the development work products with the defined quality objectives and acceptance criteria.

In Figure 1, the scope of the assessment types is further refined. Please note that Figure 1 is not meant to imply a temporal sequence of activities, but rather links between activities. Figure 1 shows that assessment types take work products from different disciplines into account in order to be effective. The quality gates do not only assess the work products of one activity, but also the 'transition' between activities, like e.g. the allocation of software requirements to architectural design. Note that during requirements engineering (RE), we leverage quality gateways similar to the ones described in [18].

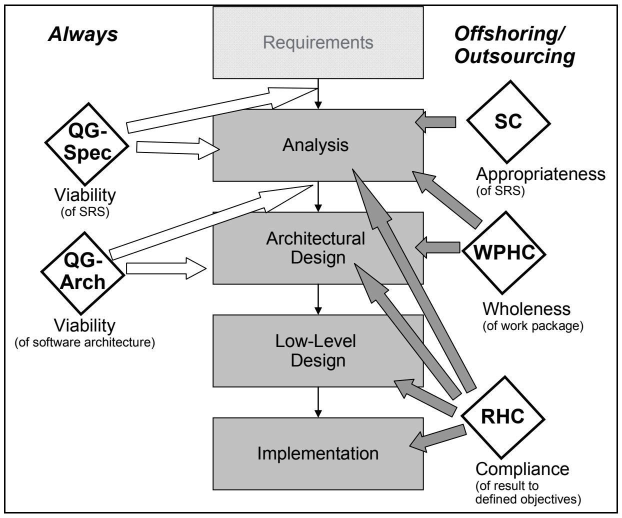

Figure 1. The "What": Scope of Assessment Types

Figure 1 depicts that the quality gates ('QG-Spec' for the specification quality gate and 'QG-Arch' for architecture quality gate) are always carried out, regardless of the shoring/sourcing model employed. The assessment types specification check ('SC') and the handover checkpoints ('WPHC' for work package handover checkpoint and 'RHC' for result handover checkpoint) are usually only employed in projects where a considerable part of the development is carried out by external suppliers or the offshore development centres. This is actually an aspect of the framework tailoring and will be explained in more detail later.

In this paper, we focus on the assessments within the disciplines 'Analysis', 'Architectural Design', 'Low-Level Design' and 'Implementation' (which is why the requirements box is greyed out in Figure 1). The discussion of failure-based quality assurance approaches like 'testing' is out of scope of this paper. Please note that Figure 1 serves as a high-level overview, and thus only gives a simplified view: For example, we left out the link between 'Requirements' and 'Architectural Design' activities or the fact that we re-assess the SRS in the architecture quality gate (QG-Arch).

According to the different goals we pursue with the assessment types, the concrete assessments are carried out at different points in time. Figure 2 delineates the points in time when the according assessments are usually carried out within the Rational Unified Process [19]. The first work packages (WP) are implemented onshore, while subsequent work packages are implemented mainly by the offshore team. It also shows that the work package handover checkpoint is distinguished in two parts: The initial high-level check WPHC(A) is to assure that all 'production means' are in place to start the implementation of work packages. The lower-level check WPHC(B) is applied to the single work packages and assures that they are 'whole' from the development team point of view. This distinction will be explained further in a later section. The bowed arrow between the handover checkpoints indicates that we carry out tool-based automatic code quality assurance checks during implementation.

Please note that the exact points in time of all assessment types can vary from project to project.

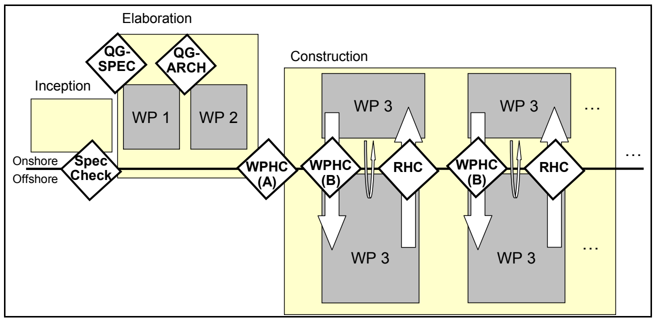

Figure 2. The "When": Assessments in a 'Standard' Offshore-Project Live-Cycle

In the standard offshore project setting, the shoring-specific assessments (SC, WPHC and RHC) are carried out jointly with the offshore team members. In Figure 2, this is depicted by placing the according assessments on the central horizontal line between the onshore and offshore activities. For large or high-risk work packages, we advice to collocate the relevant offshore and onshore team members to carry out handover checkpoints. For less critical work packages, using video conferences and other collaboration tools usually suffice. On the other hand, quality gates are mainly conducted onshore. However, at Capgemini sd&m we follow the best practice that the offshore analysts and architects work onshore during the early phases of the project live cycle (see also [20]), so they can take part at the quality gates.

After having discussed the high-level view on the overall assessment framework, we present its three constituent parts in more detail.

3. SPECIFICATION CHECK

When software requirements specifications (SRS) are written onshore and implemented offshore or when the implementation is outsourced, a couple of new challenges arise.

- The terminology and the understanding of core concepts can differ substantially [10, 11], which is considered a major challenge in GSD [21]. A more detailed SRS can be necessary from the offshore/outsourcing development team point of view due to heterogeneous understanding of requirements [10-12].
- The structure of a SRS can hamper effective knowledge transfer. Over-structuring a SRS as well as using a completely flat SRS should be avoided.
- The tracing model can be insufficient from the offshore developer perspective (e.g., missing links between SRS

artifacts), as well as from the management perspective (e.g., not allowing impact analysis). It is well known, that traceability plays a crucial role in GSD [22].

Within the specification check (SC), these issues are assessed jointly by the offshore business analysts and architects:

Check of definition of core concepts and terminology used to structure the SRS. The reference model against which we check the precision of the used concepts and terminology is the Capgemini sd&m specification method [16]. It is based on a precise meta-model and as such defines all constituent parts of a standard SRS (examples of such parts are 'use case', 'dialog', 'batch'). Deviations from the specification methodology are allowed but must be motivated reasonably. Together with the offshore business analyst and architect, it is determined whether the structure of the SRS will support a precise and detailed specification, such that the offshore development team will be able to implement it.

Check that the SRS will enable effective knowledge transfer. Over-structuring specifications can lead to a pigeonhole effect, where information is split at a too fine grained level and scattered over the whole SRS. This makes it difficult to restore the overall context of information. Table 2 shows two example questions from the checklist used in the SC to investigate this issue.

Table 2. Example Questions from SC (shortened)

Question	Benchmark	Rational
Who will have to read which parts of the specification to get all the information he needs to do his duty?	The concept should be structured in a way that not everyone has to read everything. Reading instructions should be provided.	If everyone has to read everything to do his duty, then the concept is structured in the wrong way.
Will you provide introductory chapters, outlining the business environment of the customer? Will the concept contain an overview of the specified business logic?	Provide introductory chapters for an easy access to the topic. Consider that the concept will also be read by people, that haven't talked to the customer and only have a limited understanding of the underlying domain (e.g. German tax law).	The business analyst at the offshore/ outsourcing site must understand what's happening in the customers business. Only then he will be able to answer questions to the remote team.
…	…	…

Check of the employed tracing model. In GSD project management is more difficult than in co-located project settings due to decreased insight into status [15]. The employed tracing model must thus effectively support project management and controlling. Besides traceability between artefacts on different levels of abstraction, traces between artefacts on the same level of abstraction are important from the developer point of view. For example, a mapping of the domain entities as referenced in use cases onto the data model might be necessary.
In summary, the SC ensures the appropriateness of the information model and the structure of the SRS in the sense, that both, the onshore and the offshore team members will be able to efficiently work with it.

Related work and contribution: Conveying requirements to remote teams is commonly regarded as one of the prime challenge in GSD, where different cultural and engineering backgrounds can lead to different interpretations of requirements [10-12].

Some authors concluded that in GSD, a SRS has to be more detailed [13, p. 146ff] than in co-located settings. Reviews of the SRS committed by the receiving team have been proposed [23, p. 147]. We go a step further and propose not only to evaluate the level of detail of SRS, but to *introduce an early check where the structure of the SRS is assessed for 'offshore-fitness' – jointly by the onshore and offshore/outsourcing team*. This ensures that all stakeholders will eventually be able to use the SRS to fulfil their respective duties. Laveraging a check like the SC seems to be especially valuable in offshore outsourcing contexts in order to align the different terminology used within different companies.

4. QUALITY GATES

At Capgemini sd&m, quality gates are comprehensive assessments executed at specific points in time assessing the maturity and viability of produced artefacts (milestones) as well as the processes which are used to produce them. We have developed quality gates for securing the quality of the software requirements specification and the software architecture. With our quality gates, we pursue the following high-level objectives [17]:

- To make the maturity of core work products and processes transparent.
- To derive effective countermeasures for major problems encountered.
- To 'standardize' in-depth assessments to a reasonable extent.

One main characteristic of the quality gates is their focus on in-depth examination of content instead of formal checks. We do not only check the existence of a component model or the existence of a change management process but we concentrate on checking their appropriateness for the given project context.

The quality gate concept at Capgemini sd&m consists of two main processes. First, the 'quality gate application process' addresses all operative aspects of carrying out quality gates within one specific project. Second, the 'continuous improvement process' is used to constantly sharp the quality gates. The main goal is to collect the lessons learned from all applied quality gates, distil potential improvements and feed them back in the form of adaptations of the quality gates. This process is owned by the quality gates manager, a company wide role.

In former work, we described the overall quality gates concept in more detail (see, e.g. [16] and [17] for overviews).

The quality gates are devised in a modular fashion based on 'evaluation steps'. The goal we pursue with this modularization is to concentrate on one quality aspect at a time. This helps to efficiently achieve a high assessment coverage. Table 3 and Table 4 list the assessment steps for the QG-Spec and QG-Arch respectively.

Table 3. Evaluation steps of QG-Spec

Step	Evaluation Object	Evaluation Perspective	Tools/ Methods
1	SRS, tracing matrices	Structure (compliance to specification method)	Checklists/ Change scenarios
2	SRS, tracing matrices	Allocation of existing requirements, tracing	Checklists/ Change scenarios
3	SRS	Feasibility of SRS	Checklist with specification anti-pattern
4	SRS	Comprehensibility	Checklists/ Walkthrough
5	SRS	Structuring of business logic	Checklists
6	n/a	Focal points from previous steps	No specific method
7	Process descriptions	Appropriateness of employed processes	Checklists

The evaluation steps concentrate on one perspective at a time.

Table 4. Evaluation steps of QG-Arch

Step	Evaluation Object	Evaluation Perspective	Tools/ Methods
1	SRS	Appropriate continuation of SRS since QG-Spec	Checklists/
2	Architecture documentation / Coded application component interfaces	Conformance to the Capgemini sd&m architecture standard. Usage of architecture good practices	Checklists, Software Cockpit
3	Architecture documentation	Alignment of architectural design decisions with business drivers. Suitable tradeoffs between architectural design decisions	"Lightweight" Architecture Tradeoff Analysis Method [17]
4	Proof-of-concept/ technical prototype/ coding templates	Proof-of-concept reliable, prototype viable and templates of high quality and suitable for wide-scale use	Checklist, Automatic Tools
5	n/a	Focal points from previous steps	No specific method
6	Process descriptions	Appropriateness of employed processes	Checklists

Each evaluation step of our quality gates has its own usage concepts with a detailed description on how to apply the tools and methods, as well as entry and exit conditions. The steps can thus be used separately used and combined with each other. This supports the tailoring of the quality gates to specific project settings. For example, in the presence of disagreeing customer stakeholders, it is reasonable to include evaluation step three of the QG-Arch (lightweight-ATAM) into a QG-Spec: We sometimes encounter conflicting goals between the IT-department (responsible for eventually operating the system) and the eventual users. In such a situation, an early ATAM session can help to find an overall prioritization of the requirements and to resolve lurking conflicts, before the SRS is finally signed-off.

Related work and contribution: Much valuable research on inspections has been accomplished. However, we already argued that most inspection techniques focus on the evaluation of work products, leaving out the assessment of relevant processes [17]. Moreover, in GSD it is especially important that clear processes are defined and actually lived within the projects daily live (see, e.g., [20]). *The quality gates encompass a specific evaluation step to carefully assess the relevant processes*. Usually, inspections are devised either for assessing a SRS or a software architecture. *The 'construction kit'-approach allows to assemble the evaluation steps of different quality gates in order to cater for different project settings. This seems to be an interesting new trait in software inspections research.*

5. WORK PACKAGES AND HANDOVER CHECKPOINTS

The global distribution of the software development processes over organizational and geographical borders poses new risks and intensifies existing challenges to project management as compared to collocated software development [5-7, 23].

Using clearly defied work packages and synchronizing their handover with checkpoints have been proposed by other researches [5, 13] and ourselves [24] in order to tackle these problems. We observed however, that the definitions of 'work package' and 'handovers' thereof must reflect the employed shoring and sourcing models in order to be effective. It is beneficial to use the idea of 'tailoring' work packages and handovers according to the project setting.

Basically, our work packages contain three types of information (see also [24]):

- **Specification-information:** Cohesive subsets of the SRS and test case specifications.
- **Design-information:** external interfaces of the modules (i.e., the external technical view on the subsystem which realizes the conceptual component).

- **Management-information:** The list of artifacts which are bundled into the work package, the schedule for the work package, and the definition of quality objectives and acceptance criteria for the work units.

Before construction of a work package can start, it has to pass the 'Work Package Handover Checkpoint' (WPHC). The result of work package construction is checked in the 'Result Handover Checkpoint' (RHC). In between these two checkpoints, comprehensive code quality assurance is applied, like continuous integrations, systematic code reviews, and automatic checks for architecture compliance. The gathered metrics are aggregated and displayed in the 'Software Cockpit', our standard project quality dashboard.

To gain a positive outcome of the WPHC the onsite and offshore team members must agree on the 'transfer' of the work package: The offshore/outsourcing developers declare the wholeness of the work package in the sense that they have enough information in order to implement the work package.

The WPHC encompasses two kinds of evaluations: WPHC(A) is the initial pre-construction check and ensures that the 'production means' are in place and mature (*wholeness* of production means). This check is not applied to single work packages, but to core work products, processes and environments which will be used to construct all work packages. Examples of such core work products are specifications of cross-cutting concepts like authorization or multi-tenancy. Examples of core processes are change management or configuration management. Examples of core environments are ready build-servers or common setup of IDEs.

WPHC(B) is the actual work package check which ensures that implementation of a work package can safely start (*wholeness* of single work packages). This check is applied to each single work package and ensures the wholeness of the contained specification, design and management information.

In turn, the RHC is passed if all team members agree on transferring the result of the work package implementation to the integration team. Here, it is determined whether the result has been completed according to the defined quality objectives (*compliance* of work package result). This is also the point, where planned and actual effort of work package implementation is compared and feed back to project management.

The notion or work packages along with the handover checkpoints offer the following benefits:

- *Effective delegation of work* and responsibility, and controlling of compliance to clearly defined acceptance criteria
- *Transparency of project status* at all times and continuous progress tracking
- *Synchronization of remote teams* at clearly defined handover points
- *Continuously practiced one-team-spirit* and common view on the customer requirements

The specification- and architecture methods used at Capgemini sd&m support the definition of self contained work packages based on cohesive and loosely coupled components. This decreases communication costs between distributed teams. Work packages allow to increase the responsibility of the offshore team and to avoid remote micro-management. The handover checkpoints provide a rigorous yet flexible means to make work package progress transparent and assure the wholeness of work packages and high quality development results. Especially the work package handover checkpoints are applied jointly by the onsite and the offshore team. This fosters the one-team spirit and ensures effective knowledge transfer.

Work packages allow effective earned value analysis on a granularity which is meaningful for the project management. Metrics are aggregated on work package level. The gathering of metrics is done automatically to a large extend, and the results are displayed in the Software Cockpit. Sample metrics are test case coverage, number of failure of test cases, number of code reviews completed, development status of artefacts, number of review defects found, as well as design metrics and architecture compliance metrics. The Software Cockpit also provides means to carry out quality forecasts and quality trend analysis. It thus provides the project management comprehensive 'product based' insights based on the quality and maturity of work packages. During work package construction, the metrics and trend analysis within the software cockpit guides the offshore development team and the chief architects.

To plan the day-to-day development, 'coarse grained' work packages are usually broken down into more manageable 'fine grained' work packages which can be assigned to single developers. The work breakdown of coarse grained into fine grained work packages as well as planning, staffing and scheduling of the fine grained work packages is best carried out by the receiving team.

Of course, the size of transferred work packages depends on various parameters. At the early stages of a project, a work package contains rather few specification artifacts, like a few use cases plus according business rules and domain objects (as indicated by the dashed rectangle No. 3 of Figure 3).

In early stages the onshore team usually devises the interfaces (denoted as 'IF' in Figure 3) of the according application component, as well as the lower level design of the component internals. In this case, a work package then contains a few use cases, domain objects and business rules (dashed rectangle No. 3 in Figure 3), the high-level design (dashed rectangle No. 2) and the low-level design (dashed rectangle No. 1). In the following, we thus call this the '1-2-3'-model.

Gradually transferring more and more responsibility to the offshore team, the '2-3'-model is employed where the offshore team creates the lower level design itself. As soon as the development process is stable enough, the '4'-model is leveraged: Here, a whole conceptual component bundling several docent use cases, business rules and domain objects can be transferred. The offshore team will then subdivide the conceptual component into application components (via defining component interfaces, IFs), and creates all the lower level design.

Please note that Figure 3 is only an informal and very much simplified illustration of the notion of work packages and the underlying development artifact model.

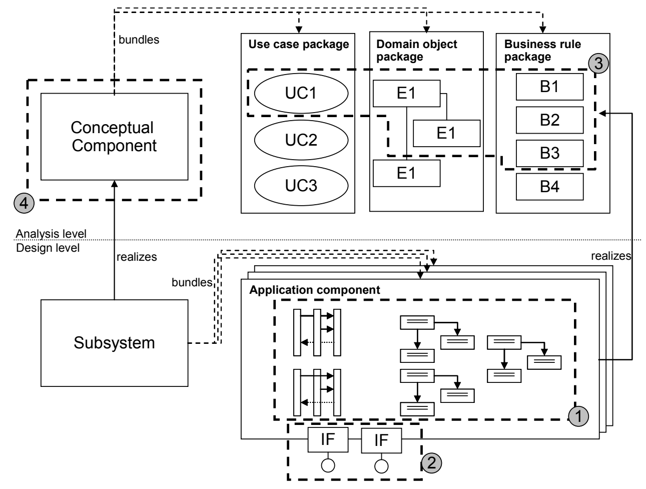

Figure 3. Work Package Ingredients (illustrative)

Not shown in Figure 3 are the management artefacts nor the functional test cases, which are nevertheless core parts of a work package. We observed, that functional test cases effectively complement the SRS in that they provide further context on how the system will work.

We note that even a precise notion of work packages do not at once and for all solve the problem of task assignment in distributed work settings (see also [10]). However, it helps to scrutinize all kinds of interdependencies (functional dependencies and temporal dependencies). Functional dependencies can then be tracked within tracing matrices, while temporal dependencies are modelled within the project plan. Please also note that although the temporal responsibility to perform certain activities or tasks on a work package is transferred between shores, we mandate that ownership (accountability) always resides at the onshore team (and most often one person).

Related work and contribution:

Industrial research shows the need of clear notions for work packages and handover checkpoints to manage GSD: A study conducted at Siemens reports that 25% of the encountered release defects had their origin in incomplete definitions of work packages [14].

Another study at Philips revealed that acceptance procedures of deliverables where unclearly defined in up to 80% of the projects, which entailed integration test budget overruns [7]. With our handover checkpoints, we address the stated need for frequent collaborative reviews [14].

Empirical research at Capgemini has shown that formal and informal mutual adjustment is of crucial importance for GSD project success [15]. The consequent introduction of handover checkpoints clearly enforces formal mutual adjustment. However, we want to exemplify another key observation: *The handover checkpoints also triggerd intensive informal mutual adjustment, especially between remote team members that did not had direct contact with each other before.* When they where 'forced' to collaborate within a handover checkpoint, they observed that 'those other guys are not that bad'. Quickly they noted the competence of their team mates which fostered mutual respect, trust and intensive liaisons.

In the next section, we describe, how the handover checkpoints can be tailored in order to serve for different shoring and sourcing settings.

6. FRAMEWORK TAILORING

From our experience, single project settings are very different from each other such that an overly complicated tailoring guideline might not be handled effectively in practice. We thus promote a rough but easy to use categorization on how to best leverage the assessment framework in specific project settings. It then lays in the responsibility of the project to further adapt the assessment framework to its specific needs.

Basically, the 'tailoring' of the assessment framework is done on two levels:

1) **Tailoring on framework level**: This amounts to using or not using an assessment type within a specific project setting.

2) **Tailoring on assessment type level**: This means to adapt a single assessment type to the project setting at hand.

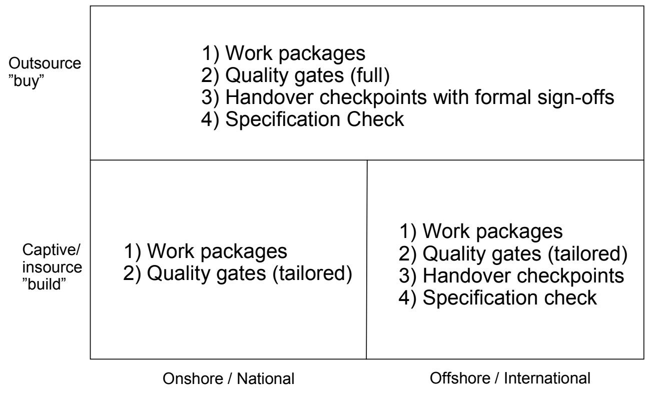

Figure 4. Framework-Level Tailoring

1) Tailoring on framework level: The tailoring on framework level is illustrated in Figure 4.

The key points to observe in this figure are the following:

Knowledge transfer challenges increase when development work is distributed [13]. As the specification check and the handover checkpoints support effective knowledge transfer, these two techniques are introduced when moving from onshore intra-organisational settings to offshore settings. Further, one main goal of the specification check is to establish a common understanding of core specification concepts (like 'use case' or 'business rule'). As the Capgemini specification methodology is used throughout Capgemini sd&m, this issue is less critical in pure onshore intra-organisational projects. These are the main reasons why we use the specification check and the handover checkpoints mainly in projects with considerable offshoring and/or outsourcing of work.

Furthermore, outsourcing can introduce tough legal and contractual issues. Loss of control in project management and varying development standards [25] as well as diverging long term interests can make it necessary to tighten handover checkpoints by formal sign-offs, when moving to outsourcing scenarios. This observation holds true in all outsourcing settings, regardless whether offshoring is involved or not. Finally, SRS and software architectures have to be extremely stable in outsourcing contexts due to the highly formal and therefore effort consuming change management processes. For this reason we advice to apply always all evaluation steps of the quality gates and discourage pruning them when working in outsourcing contexts.

2) Tailoring on assessment type level. Finer grained tailoring is applied only to quality gates and the handover checkpoints. The specification check is not tailored. As already mentioned, it is used in offshore/outsourcing projects and not used in pure onshore intra-organisational projects. In chapter 4, we already sketched how quality gates can be tailored to specific project needs. In the following, we thus concentrate on the tailoring of the handover checkpoints. Basically, we distinguish between three different models of how to employ the handover checkpoints. While the first two models are used at different stages in offshore projects, the third model is used in outsourcing contexts.

1. 'Teamed Design' Model

This model (as illustrated in Figure 5) is usually employed in offshore projects for the first couple of work packages (WP1, WP2 in the illustration) in the overall lifecycle.

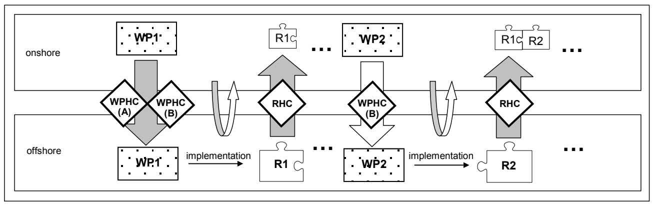

Figure 5. 'Teamed Design' Model

The basic idea is to devise as much of the work package design as possible together. For this reason, we call this model 'teamed design' model. The characteristics of the handover checkpoints in this model are:

- All handover checkpoints carried out jointly by the remote teams.
- Most often used with work package model '1-2-3' or '2-3', as described in section 5.

The common quality dashboard for continuous code quality control (indicated by the bowed arrows in between the handovers) is used by onshore and offshore colleagues alike. The integration of application components or subsystems (denoted R1 and R2 in the illustration) is usually done onshore.

2. 'Experienced Offshore Team' Model

As the projects moves on, the offshore development team gains more and more experience in designing solutions, whence more and more responsibility is shifted to the offshore team members.

In particular, the high-level design and the low-level design are primarily done offshore, although the onshore team can still participate and support this activity. This is also the best model to be applied within a well established maintenance process, where the offshore team works on complex change requests.

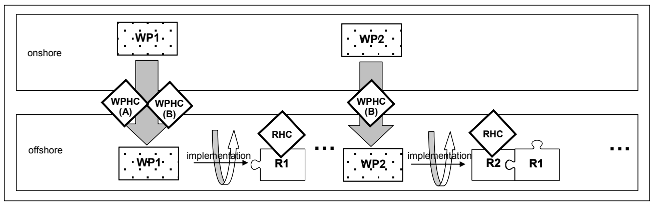

Figure 6. 'Experienced Team' Model

Characteristics of this model as sketched in Figure 6 are:

- The work package handover checkpoints (WPHC) are still carried out jointly by onshore and offshore teams, but the result handover checkpoints (RHC) is applied by offshore team.
- The quality dashboard is manly used by the offshore development team which reports aggregated key quality indicators to the onshore project manager and architects.

In this model, the integration of application components or subsystems is mainly done offshore. Still, much of the functional end-to-end testing is carried out onshore. This handover checkpoint model works well with the work package model '2-3' and model '4'.

3. 'Outsourcing' Model

The main difference between the outsourcing model and the two proceeding models is that all handover checkpoints are carried out by the onsite team, as illustrated in Figure 7.

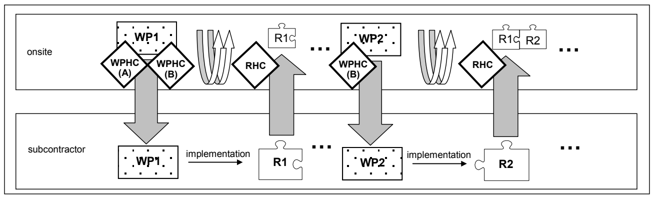

Figure 7. 'Outsourcing' Model

The characteristics of this model are:

- All handover checkpoints are carried out by the onsite team.
- Rigorous use of the quality dashboard for continuous code quality control is heavily leveraged by the onsite team (as indicated by the multiplicity of the bowed arrows).

The integration of application components or subsystems is done onsite.

Related work and contribution:

Different models of work assignment (work packages) and the use of synchronisation points like our handover checkpoints have been discussed by various researchers [5, 13]. However, the descriptions sometimes lack the detail to serve as a starting point for the industrial practice.

What is missing is a clear description in which way different 'work package models' and 'handover models' play together, and which models are suitable for which offshore or outsourcing setting.

This paper explained different work package models and models for handover checkpoints used at Capgemini sd&m, as well as their suitability for different project scenarios. It thus contributes to solve the challenges of effective handovers, which has recently be coined by Berry Boehm as one of the key future challenges in GSD [26].

7. CONCLUSION AND FUTURE WORK

This work summarizes the outcome of a two years research initiative carried out at Capgemini sd&m. The initiative started from the observation that during the transition from requirements engineering to software design and implementation, one encounters four major challenges:

- To ensure the *appropriateness* of software requirement specifications (for all involved stakeholders)
- To ensure the *viability* of software requirement specifications and software architectures
- To ensure the *wholeness* of work packages from the developer point of view
- To ensure the *compliance* of the development work products to predefined quality objectives and acceptance criteria.

We presented an assessment framework consisting of three 'assessment types' and argued that this framework effectively addresses the four challenges while taking the specifics of global software development (GSD) into account:

- The 'Specification Check' surfaces differing understanding of core concepts in software requirement specifications (SRS) and evaluates, whether the structure of the SRS is appropriate for the different stakeholders.
- The 'Quality Gates' evaluate the viability of two core work products: SRS and software architectures. Flaws in these work products are major risks, especially in GSD where resolving such flaws is particularly difficult due to the impeded communication.
- The 'Work Package Handover Checkpoints' ensure the wholeness of work packages and guarantees sufficient knowledge transfer such that the offshore developers can safely start their work. In a complementary fashion, the 'Result Handover Checkpoints' ensure that the development result actually complies to the defined quality objectives and acceptance criteria, and that the result has been developed in the planned time and budget. These two handover checkpoints help to orchestrate the communication and collaboration in GSD projects.

We also showed how the framework can be tailored to different development models supporting offshore development as well as outsourcing of work to third party suppliers. The assessment framework is currently adopted company-wide for our large GSD projects.

Besides devising the assessment framework, we drew some general lessons learned during this two years initiative:

We experienced that informal communication between peers seems to be of paramount importance in GSD: It is necessary to quickly cope with situations which are not covered by the formal communication techniques, like processes, guidelines and policies. Interestingly we observed, that formal communication techniques like our handover checkpoints also stipulated lots of informal communication: Once the onshore and offshore colleagues carried out a handover checkpoint, the first step towards intensive collaboration was done. Many informal liaisons between peers originated from the formal handover checkpoints and repeated successful handovers deepened the trust between offshore and onshore peers.

Probably the most important lesson learned by this two years initiative is the insight that global GSD necessitates a deep change in a software engineers work habits: In a flat world, embracing proactive and continuous adjustment to different engineering backgrounds becomes paramount (see also [21]).

Though the presented assessment framework is gradually becoming a standard at Capgemini sd&m, some work is left to be done:

- Further differentiation of project settings and the appropriate tailoring of the assessment framework could be fruitful. Appropriate research questions could thus be: *a) Is a further differentiation between intra-national, near- and farshore projects reasonable? What effects have the different 'magnitudes' of geographical distribution on assessment-based quality assurance techniques? b) Is a further differentiation between onshore outsourcing and offshore outsourcing reasonable? What are the precise differences between onshore outsourcing and offshore outsourcing with respect to quality assurance?*
- In former work, we found some weaknesses of inspections with respect to assessing the correctness and completeness of SRS [27]. This issue is compounded in GSD, as turnaround times for clarification of such flaws increases due to impeded communication. From an industrial perspective, a highly relevant research question would thus be: *"Is it more reasonable to invest into preventive or failure-based quality assurance techniques instead of inspections in order to increase the correctness and completeness of SRS?"* Further research is needed to clarify this question in general, but especially in global software development.

8. REFERENCES

[1] The Standish Group International Inc. 2003. CHAOS Chronicels v3.0. West Yarmouth.

[2] Degner, C. and Olsson T. 2005. Quality Assurance in Requirements Engineering. Engineering and Managing Software Requirements. Springer. Berlin-Heidelberg.

[3] Clements, P., Kazman, R. and Klein, P. 2002. Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley Longman Publishing Co., Inc.

[4] Boehm, B., W. 1981. Software Engineering Economics. Prentice-Hall.

[5] Cusumano, M. A. 2008. Managing software development in globally distributed teams. *Commun. ACM* 51, 2 (Feb. 2008), 15-17. DOI=http://doi.acm.org/10.1145/1314215.1314218

[6] Cusick, J. and Prasad, A. 2006. A Practical Management and Engineering Approach to Offshore Collaboration. IEEE Softw. 23, 5 (Sep. 2006), 20-29. DOI= http://dx.doi.org/10.1109/MS.2006.118

[7] Kommeren, R. and Parviainen, P. 2007. Philips experiences in global distributed software development. Empirical Softw. Engg. 12, 6 (Dec. 2007), 647-660. DOI= http://dx.doi.org/10.1007/s10664-007-9047-3

[8] William A., Frank M. and Vardi, M., Y (Eds.). 2006. Globalization and Offshoring of Software. ACM task force. ACM.

[9] Herbsleb, J. D. 2007. Global Software Engineering: The Future of Socio-technical Coordination. In 2007 Future of Software Engineering (May 23 - 25, 2007). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 188-198. DOI= http://dx.doi.org/10.1109/FOSE.2007.11

[10] Herbsleb, J. D., Paulish, D. J., and Bass, M. 2005. Global software development at siemens: experience from nine projects. In Proceedings of the 27th international Conference on Software Engineering (St. Louis, MO, USA, May 15 - 21, 2005). ICSE '05. ACM, New York, NY, 524-533. DOI= http://doi.acm.org/10.1145/1062455.1062550

[11] Ebert, C. and De Neve, P. 2001. Surviving Global Software Development. IEEE Softw. 18, 2 (Mar. 2001), 62-69. DOI= http://dx.doi.org/10.1109/52.914748

[12] Herbsleb, J. D. and Mockus, A. 2003. An Empirical Study of Speed and Communication in Globally Distributed Software Development. IEEE Trans. Softw. Eng. 29, 6 (Jun. 2003), 481-494. DOI= http://dx.doi.org/10.1109/TSE.2003.1205177

[13] Hussey, J., M. and Hall, S., E. 2007. Managing Global Development Risks. Auerbach Publications.

[14] Berenbach, B. 2006. Impact of organizational structure on distributed requirements engineering processes: lessons learned. In Proceedings of the 2006 international Workshop on Global Software Development For the Practitioner (Shanghai, China, May 23 - 23, 2006). GSD '06. ACM, New York, NY, 15-19. DOI= http://doi.acm.org/10.1145/1138506.1138511

[15] Fabriek, M., van den Brand, M., Brinkkemper, S., Harmsen, F. and Helms, R. 2007. Improving offshore communication by choosing the right coordination strategy. Technical Report UU-CS-2007-021. Department of Information and Computing Sciences. Utrecht University.

[16] Salger, F., Sauer, S., and Engels, G. 2009. Integrated specification and quality assurance for large business information systems. In Proceeding of the 2nd Annual Conference on india Software Engineering Conference (Pune, India, February 23 - 26, 2009). ISEC '09. ACM, New York, NY, 129-130. DOI= http://doi.acm.org/10.1145/1506216.1506242

[17] Salger, F., Bennicke, M., Engels, G., and Lewerentz, C. 2008. Comprehensive Architecture Evaluation and Management in Large Software-Systems. In Proceedings of the 4th international Conference on Quality of Software-Architectures: Models and Architectures (Karlsruhe, Germany, October 14 - 17, 2008). S. Becker, F. Plasil, and R. Reussner, Eds. Lecture Notes In Computer Science, vol. 5281. Springer-Verlag, Berlin, Heidelberg, 205-219. DOI= http://dx.doi.org/10.1007/978-3-540-87879-7_13

[18] Robertson, S. and Robertson, J. 2006. Mastering the requirements process (2nd Ed.), Addison-Wesley Longman, Amsterdam.

[19] IBM Corporation. Rational Unified Process. 2007. Version 7.0.1.

[20] Clerc, V. 2008. Towards architectural knowledge management practices for global software development. In Proceedings of the 3rd international Workshop on Sharing and Reusing Architectural Knowledge (Leipzig, Germany, May 13 - 13, 2008). SHARK '08. ACM, New York, NY, 23-28. DOI=http://doi.acm.org/10.1145/1370062.1370068

[21] Bartelt, C., Broy, M., Herrmann, C., Knauss, E., Kuhrmann, M., Rausch, A., Rumpe, B., and Schneider, K. 2009. Orchestration of Global Software Engineering Projects - Position Paper. In Proceedings of the 2009 Fourth IEEE international Conference on Global Software Engineering - Volume 00 (July 13 - 16, 2009). ICGSE. IEEE Computer Society, Washington, DC, 332-337. DOI= http://dx.doi.org/10.1109/ICGSE.2009.52

[22] Lormans, M., Dijk, H. v., Deursen, A. v., Nocker, E., and Zeeuw, A. d. 2004. Managing Evolving Requirements in an Outsourcing Context: An Industrial Experience Report. In Proceedings of the Principles of Software Evolution, 7th international Workshop (September 06 - 07, 2004). IWPSE. IEEE Computer Society, Washington, DC, 149-158. DOI= http://dx.doi.org/10.1109/IWPSE.2004.14

[23] Sangwan, R., Bass, M., Mullick, N., Paulish, D. J., and Kazmeier, J. 2006 Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series). Auerbach Publications.

[24] Salger, F. 2009. On the Use of Handover Checkpoints to Manage the Global Software Development Process. OTM 2009 Workshops, Notes In Computer Science, vol. 5872, Springer-Verlag, Berlin, Heidelberg, 267–276

[25] Sakthivel, S. 2007. Managing risk in offshore systems development. Commun. ACM 50, 4 (Apr. 2007), 69-75. DOI= http://doi.acm.org/10.1145/1232743.1232750

[26] Boehm, B. 2006. A view of 20th and 21st century software engineering. In *Proceedings of the 28th international Conference on Software Engineering* (Shanghai, China, May 20 - 28, 2006). ICSE '06. ACM, New York, NY, 12-29. DOI= http://doi.acm.org/10.1145/1134285.1134288

[27] Salger, F., Engels, G. and Hofmann, A. 2009. Inspection Effectiveness for Different Quality Attributes of Software Requirement Specifications: An Industrial Case Study, In Proceedings of the 7th ICSE Workshop on Software Quality (WoSQ 09), IEEE Press, May 2009, 15-21, DOI= doi.ieeecomputersociety.org/10.1109/WOSQ.2009.5071542.