https://doi.org/10.48009/2_iis_2009_194-203

PROCESS MODELS AND DISTRIBUTION OF WORK IN OFFSHORING APPLICATION SOFTWARE DEVELOPMENT

Karl Kurbel, European University Viadrina Frankfurt (Oder), kurbel.bi@euv-frankfurt-o.de

ABSTRACT

Common process models for the development of application software (AS) are examined regarding their suitability for offshoring projects. Such projects require communication and interaction among onsite and offshore project stakeholders to be well supported. Process models applied by organizations using offshore services are discussed, and a generalized offshoring life cycle model is presented. We also discuss the distribution of work between the organization that outsources AS development and the offshore organization that carries out the major share of the development work. Software development tasks performed onsite and offshore are identified.

Keywords: Offshoring, Process Model, Globally Distributed Software Development (GDSD)

INTRODUCTION

The context of this paper is the development of application software (AS) or more specifically, business information systems (IS) by or for organizations in Western European countries and the US. We take an approved project proposal (i.e. the management decisions to launch the project and to build the system have been made) as the starting point for the development project, and discuss various aspects of process models with regard to offshoring.

In the next section, the characteristics of common process models for AS development are outlined and the requirements resulting from the fact that parts of the AS development process are offshored are discussed. This is followed by a brief evaluation of common process models and their suitability for development projects which involve offshoring. Offshoring-specific process models found in practice are described in the next section. Based on the findings from this analysis, a generic process model for offshoring projects is presented. Some conclusions are drawn and related issues are addressed in the final section.

STATE-OF-THE ART IN THE FIELD OF PROCESS MODELS

Any development effort needs a specification of the problems to be tackled. Starting from such a problem specification, there are many ways to conduct the development project. *Software process models* help to arrange development activities into a specific order. A software process model is an ordered set of activities with associated results which is followed during the production and evolution of software. It is an abstract representation of a type of software process.

A large number of software process models have been proposed since the beginning of software engineering, categorized in many ways, and described by attributes such as:

linear vs. iterative development,
sequential vs. incremental development,
plan-driven vs. agile development,
model-driven vs. evolutionary development.

Since software development costs are significantly lower in Asian, Latin American and Eastern European countries than in the United States and Western Europe, many software orders have gone to vendors in such countries. While the price has been the driving factor for many years, other reasons such as access to qualified personnel and increasing the speed to market have also emerged [10].

AS development projects that lend themselves easily to offshoring are projects that automate well-documented business functions or processes where little day-to-day interaction is required. An ideal process would be one that is completely specified in terms of process steps as well as inputs and outputs of those steps. In such a case, a specification could be "thrown over the wall" (i.e. handed over to the offshore firm) and an information system would come back as the result of the project. Unfortunately most projects are not of that nature, instead they require a lot of interaction. Therefore project teams have to be set up in such a way that onsite and offshore personnel can communicate easily and at length. In addition, customer or end-user requirements need to be communicated effectively.

Sequential Process Model (Waterfall Model)

The oldest and best-understood process model is the sequential process model [15]. Since it was "the" process model for a long time, it has also been called the software life cycle model (SLC model). This model is based on the premise that the development process can be divided into distinct stages with specified inputs and outputs and well-defined results. The next stage starts when the previous one is completed. Results cascade from one stage downwards to the next stage, just like a waterfall. The flow of work is sequential and basically unidirectional as illustrated by the arrows in figure 1.

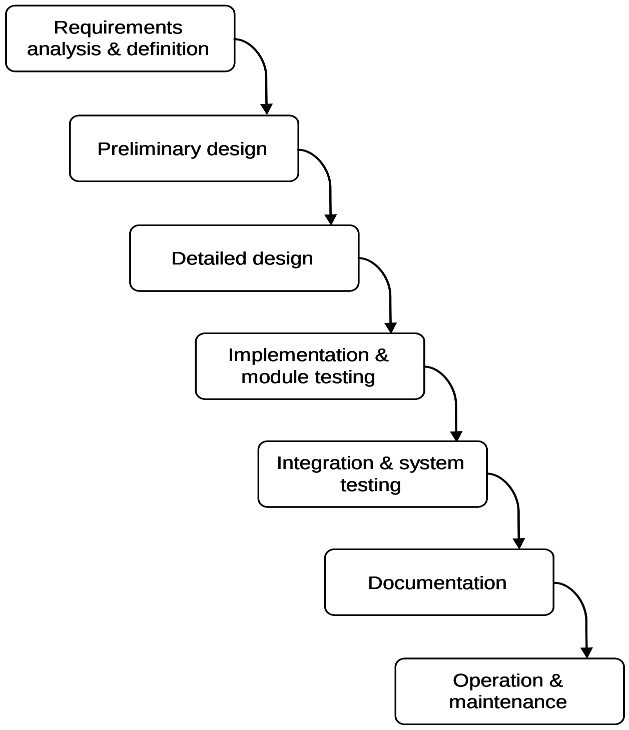
Requirements analysis & definition
Preliminary design
Detailed design
Implementation & module testing
Integration & system testing
Documentation
Operation & maintenance

Figure 1. Waterfall model

The drawbacks and advantages of the waterfall model have been extensively discussed for many years. Most authors agree that the assumption of distinct phases performed in strict sequential order does not conform to what happens in practical projects. To cope with these real-world circumstances, various modifications of the waterfall model that include revisiting earlier stages were proposed. However, even in its iterative variants, the waterfall model is mainly suited for sequential processes.

Rational Unified Process (RUP)

Evolutionary development takes into account that most systems undergo an evolution after they have been installed, i.e. they grow and change. Development cycles and prototyping play an important role in evolutionary development. Process models based on explicit development cycles supporting evolutionary and incremental processes were developed in the 1980's and 1990's both in practice and in academia (e.g. [8]). Many of these ideas were later incorporated into the *Rational unified process (RUP).*

RUP supports incremental development, dividing large projects into sub-projects. The process model is two-dimensional. The dimensions are phases and disciplines [2]. Phases extend in time, while disciplines expand into activities. The *phases* are called:

- inception,
- elaboration,
- construction,
- transition.

The *disciplines* are: business modeling, requirements, analysis & design, implementation, test, deployment, configuration & change management, project management, and environment. Disciplines extend across phases. This means that typical activities such as modeling, analysis, design, implementation and testing are not confined to one phase but are ongoing activities during the entire life cycle.

Iterations occur within phases. Iterations are intended for all phases, yet only within a phase. The result of an iteration, especially in the construction stage, is an *increment* or a subsystem which could conceivably be deployed to users as a release. The phases as such are sequential. In this way, RUP combines sequential and iterative process aspects in one process model ("serial in the large, iterative in the small" [2]).

Agile Development

An even more incremental and iterative approach is *agile development (AD)*. A famous AD document is the *agile manifesto* ("Manifesto for agile software development"). This is a position statement indicating what the authors (the "agile alliance") consider fundamental ideas for better software development [1]. Several agile methodologies have been developed. Among them are:

- Extreme programming (XP),
- Scrum (http://www.controlchaos.com/about/)
- Feature driven development (FDD – http://www.featuredrivendevelopment.com/),
- Crystal Clear [5],
- Adaptive software development (ASD – http://www.adaptivesd.com/).

Extreme programming (XP) is the best-known of the agile methodologies. It became popular through Kent Beck's book which was published in 1999 (first edition of [3]). XP is a set of simple and concrete practices that combines into an agile development process [11]. Well-known practices include close interaction with the customer, user stories, pair programming, planning game, test-driven development and refactoring [3, 11, 17].

From the description of XP, it is doubtful if extreme programming has a process model at all. In the agile community, "process" is often used with a negative undertone, implying that a process has attributes such as "disciplined" or "structured" which are disliked by agile developers. Nevertheless, the description of relationships between iterations, releases, stories and tests indicates that there is a certain process paradigm underlying XP, including iterations and prototypes ("spikes").

EVALUATION OF PROCESS MODELS FROM AN OFFSHORING PERSPECTIVE

Process models on the spectrum from one extreme – strictly sequential – to the other – completely incremental and discussion-based like XP – exhibit different requirements for project communication and interaction. Figure 2 illustrates this matter by plotting evolution intensity against the need for interaction among project stakeholders.

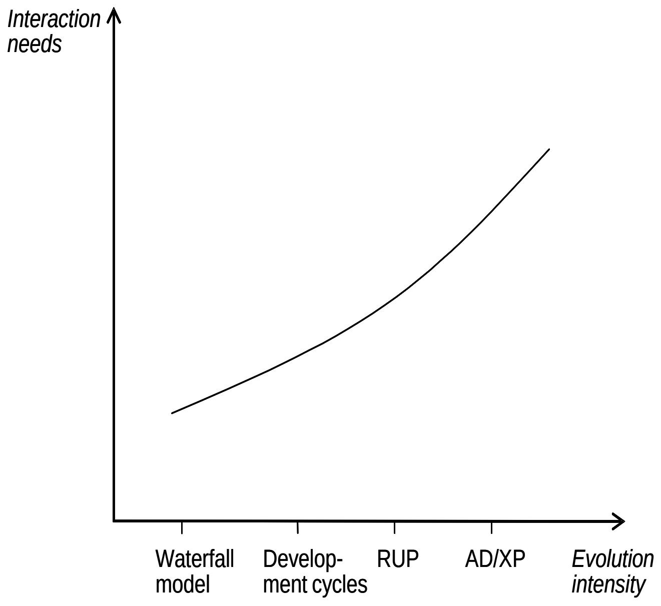

Figure 2. Spectrum of process models

In a strictly sequential process model such as the waterfall model, communication between different stakeholders is limited: System analysts talk to the customer and/or end-users in the requirements analysis & definition stage. At the end of this stage, they hand over the requirements specification to the designers. Designers or architects develop software models and give those models to the software developers at the end of the design stage, etc.

In contrast, at the other end of the spectrum, communication and interaction among the stakeholders occur continually. For example, the stakeholders (i.e. programmers, architects, quality staff, customers etc.) in XP projects meet regularly face-to-face to discuss problems, processes and possible solutions.

As stakeholders in offshoring projects are usually widely dispersed around the world, it is obviously not possible to have ongoing face-to-face discussions every other day. This is one reason why the more to the right-hand side of figure 2 a process model is located, the less likely it is that this model is used in an offshoring project. Agile approaches, for example, rely on informal processes to facilitate coordination whereas distributed software development typically relies on formal mechanisms [14].

Process models employed by offshore providers today tend to be towards the left-hand side of the spectrum (see figure 2). An obvious advantage of a sequential model is that it has well-defined milestones and documents which can be used for interfacing onsite and offshore tasks. Communication between the onsite and offshore teams is largely based on documents in this process model.

On the other hand, modern agile approaches such as XP have proved beneficial in many onshore projects. Therefore organizations are trying to transfer their benefits to offshoring projects as well. A discussion of how agile methods can be used in globally distributed software development can be found in the October 2006 issue of the Communications of the ACM (in particular [9, 14]). Although the findings, based on real-life projects, seem to be encouraging, the current state in the offshoring practice is that sequential-type process models are still dominating.

OFFSHORING SPECIFIC PROCESS MODELS AND WORK DISTRIBUTION

Organizations outsourcing AS development to an offshore provider are typically either user organizations with their own development groups, attempting to reduce costs, or software organizations developing custom information systems, with the user organization involved in the development.

The organization carrying out most of the development work is at a remote location, either far away (offshoring), or at least not close (nearshoring) so that daily face-to-face communication is not a typical characteristic. This means that communication and interaction need to be planned and ensured in a different way. The transition of work results from one organization to the other one has to be prepared based on well-defined documents, software artifacts and quality assurance.

At first sight, process models followed by organizations that offshore AS development are not much different from models for onsite development. In practice, process models are often based on one of the previously discussed models, yet with specific extensions to capture offshoring needs and reality. In particular, organizations that developed software themselves before they started offshoring are often continuing with the same process model as before. The obvious reason is that project-management experience and know-how related with that model are available.

Offshore providers, on the other hand, are somewhat limited in the choice of process model for their part, because their model has to match the offshorer's model. The looser the connections between the customer and the contractor are, the more freedom the contractor has to proceed according to their own preferences. Obviously the offshore provider cannot follow an iterative approach such as RUP if the customer's process model is strictly sequential.
Since many organizations have been applying a sequential process model or a variant of such a model in their onshore development projects, it is not surprising that sequential approaches are dominating the literature. Notwithstanding its many disadvantages, the waterfall model provides clear milestones, deliverables and points of management control that are particularly helpful when communication and feedback are by nature not as close as in an onsite project.

High-level offshore process models often exhibit the same sequential phases as a conventional process model for onsite development, i.e. business problem definition, requirements, analysis & design, implementation, testing, deployment, operations & support. The difference between various models concerns who has responsibility for which activities and in which phases, including additional activities not present in conventional process models. Who has which responsibilities depends on the chosen scope of outsourcing – i.e. does the organization wish to outsource:

1. coding and testing,
2. module design, coding and testing,
3. system design, module design, coding and testing,
4. "the problem"?

In the first and second cases, most life cycle tasks remain with the customer's project team. The fourth case, outsourcing the entire business problem, is different in that most of the responsibilities are with the offshore provider.

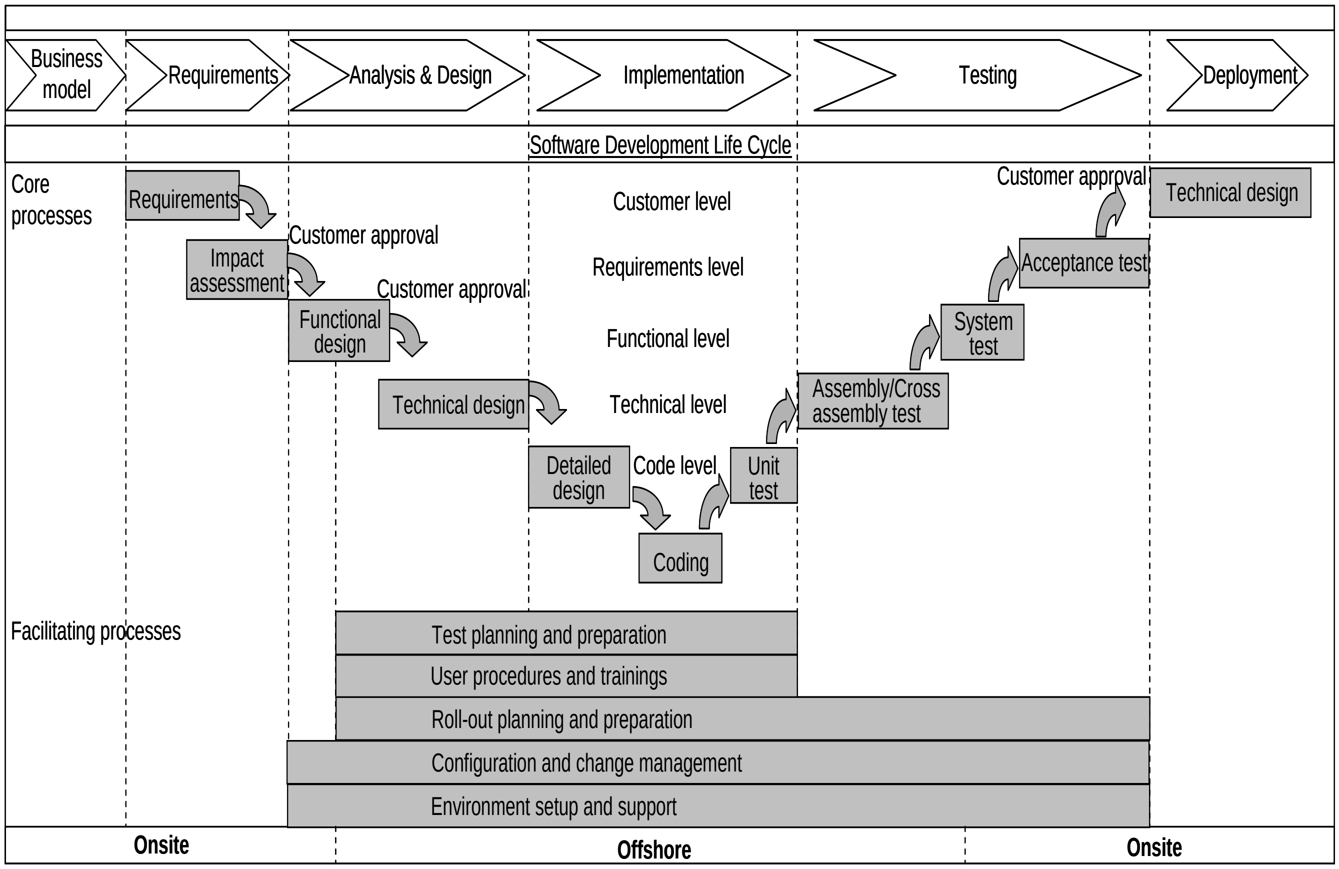

Figure 3. Onsite and offshore responsibilities in a high-level process model [16]

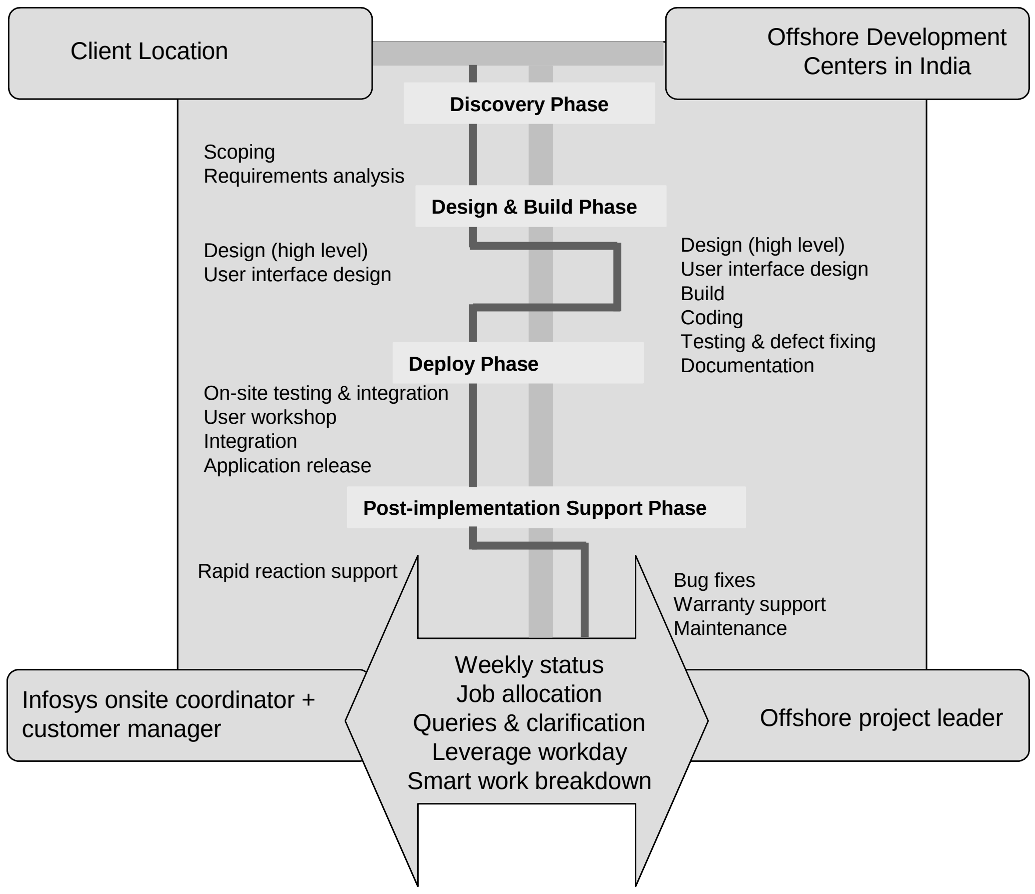

Figure 4. Infosys GDM for application software development [4]

In the third case, the cut-off stage is the system design. Often the onsite and offshore teams collaborate during the design stage as illustrated in figure 3. The diagram illustrates the high-level process model of a large, globally operating German software company. The underlying division of labor leaves project management, architecture, high-level design, risk and quality management, and the final testing responsibilities with the outsourcer while the detailed design, coding and much of the testing are the responsibility of the offshore provider.

The assignment of work packages to onsite and offshore locations is reflected in Infosys's GDM (global delivery model). Containing all major stages of the software life cycle, this model can be regarded as a process model as well. As figure 4 shows, four major phases are identified: discovery, design & build, deploy and post-implementation support. While the "discovery" (of requirements and scope) and deployment of the system happen onsite, the early design and building stages – high-level design and user interface design – involve activities on both the client's and Infosys's sides. The rest of the development stages and activities as well as bug fixing and maintenance are done in India.

By examining offshore process models in more detail, a number of additional tasks different from conventional process models can be identified. These tasks are largely due to additional communication, collaboration and control requirements between onsite and offshore personnel which are not present in onsite projects.

An example of this is illustrated in figure 5. Additional communication requirements arise from a sequence of approval steps. The process summarized in this figure is practiced by DC&M, a Brazilian offshore provider [6]. Here it is assumed that the offshore company enters the process at a very early stage, i.e. when the business problem is investigated and business requirements are being specified. The specification created by the offshore provider's consultant specifies the functional requirements from a business perspective. This specification is created in several steps and iterations, and finally approved by the customer's stakeholders.
Once formally approved, the technical specification is created. Offshore technical staff convert the functional specification into a technical solution (system design). This specification is also created in several steps and iterations with onsite and offshore technical staff involved. Finally it has to be approved by the key players from the technical perspective (e.g. offshore and onsite technical project leaders and developers).

Developing the technical solution comprises implementation and testing. An approval step has to be passed again, this time internally by the offshore provider's quality assurance staff. Afterwards the approved solution is delivered to the offshore provider's business consultant who wrote the functional specification. If the solution also passes this review step, the system is handed over to the customer for reviewing the final result with respect to the business problem and requirements that initiated the project.

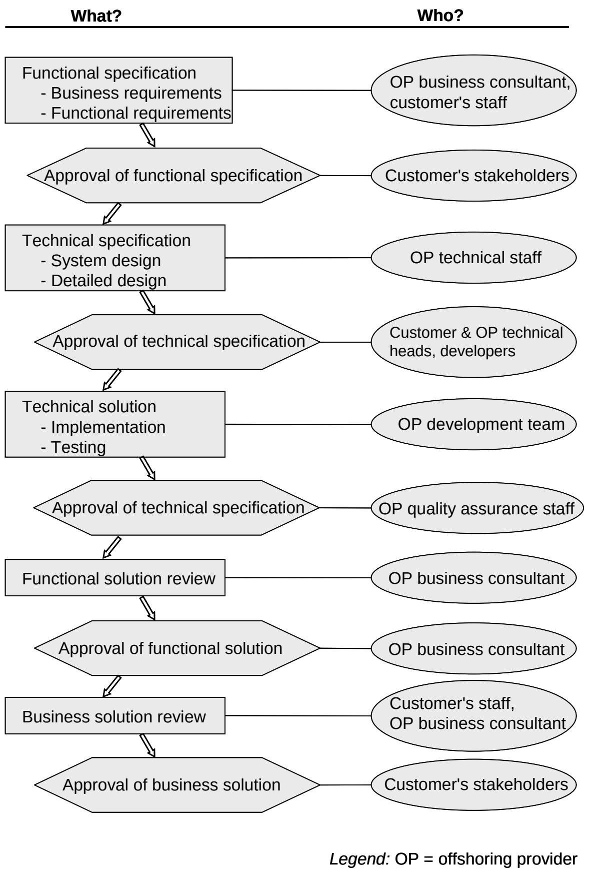
What?
Who?
Functional specification
- Business requirements
- Functional requirements
OP business consultant, customer's staff
Approval of functional specification
Customer's stakeholders
Technical specification
- System design
- Detailed design
OP technical staff
Approval of technical specification
Customer & OP technical heads, developers
Technical solution
- Implementation
- Testing
OP development team
Approval of technical specification
OP quality assurance staff
Functional solution review
OP business consultant
Approval of functional solution
OP business consultant
Business solution review
Customer's staff, OP business consultant
Approval of business solution
Customer's stakeholders
Legend: OP = offshoring provider

Figure 5. Offshoring project approval steps (based on [6])

A GENERALIZED OFFSHORING PROCESS MODEL

Based on the findings of the process model evaluations in the first section and the characteristics of offshoring process models found in practice, a generic process model for AS development which includes offshored development tasks is presented in figure 6. Taking into account that offshoring projects differ in what development tasks are outsourced to the offshore company, figure 6 attempts to capture the major stages that most projects go through. Tasks displayed in the left-hand part of figure are performed onsite, while tasks displayed on the right-hand side are performed offshore.

Depending on the development stage where the outsourcing starts, offshoring-specific tasks and stages are entered earlier or later in the project – following the business-problem specification, requirements analysis & definition, high-level design or detailed-design stage, respectively. Offshoring-specific stages are the following:

- *Examination of offshoring feasibility:* Provided that offshoring is considered a serious option, the goal is to investigate in more detail if the project is suited to offshoring.

- *Making the project ready for offshoring* includes the following tasks [12]: Determining what has to be done before a work order can be placed with the provider; setting up a project organization that takes offshoring-specific requirements into account; establishing a rough overall project schedule and a more detailed transition plan; management commitment to outsource project stages offshore; and preparing onsite project members and other stakeholders to deal with the offshoring situation.

- *Detailed offshore project feasibility:* In this stage, the customer and the offshore provider analyze in detail if it is reasonable to assume that the outsourced tasks will be solved as expected. The offshore provider needs to collect the necessary information in order to examine whether the required expertise, manpower and technical infrastructure are available or can be allocated at their site in order to be able to make a definite commitment. The customer's objective is to be sure that the partner is reliable and capable of delivering as expected. This stage includes refining the project organization, the transition plan and the project schedule.

- *Placing the offshore development order:* Provided that the offshore partner is willing and capable of performing the outsourced tasks, both parties enter into an agreement specifying among other things the work to be done, the deliverables, and perhaps process characteristics.

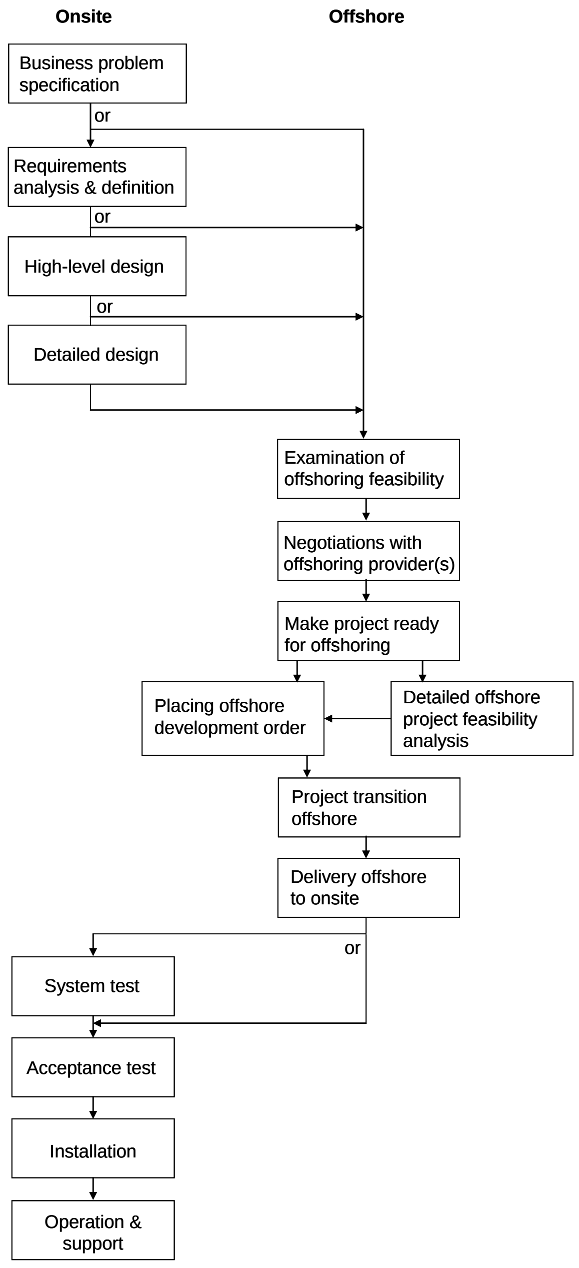
Onsite
Offshore
Business problem specification
or
Requirements analysis & definition
or
High-level design
or
Detailed design
Examination of offshoring feasibility
Negotiations with offshoring provider(s)
Make project ready for offshoring
Placing offshore development order
Detailed offshore project feasibility analysis
Project transition offshore
Delivery offshore to onsite
or
System test
Acceptance test
Installation
Operation & support

Figure 6. A generalized offshoring life cycle model

- *Project transition:* The major activities in this stage are knowledge transfer and ensuring a working project-wide technical infrastructure. The objective of this stage is to make sure that the offshore provider is able to continue the project successfully offshore. In order to acquire necessary project knowledge (e.g. domain, business-process, tool or environment knowledge), offshore personnel may have to be trained by the customer. For this purpose, employees of the offshore provider visit the customer's organization onsite in many cases, collecting information and knowledge that they take home to disseminate among their colleagues in the project.

- *Delivery:* While a significant workload will be carried by the offshore provider in the meantime, the next stage from the customer's point of view is when they get the result. This is supposed to be the working information system. In addition to the software, the system documentation, testing results and quality assurance reports will be handed over to the customer. Delivery usually takes place onsite, with offshore personnel available onsite to solve any problems detected immediately before the system goes live.

Depending on which tasks and stages were outsourced, the process continues at the customer's site with either system or acceptance testing.

Although the model in figure 6 is basically sequential, with some activities possibly going on in parallel, the offshore-specific stages need not necessarily be performed in a strict sequence nor as disjoint stages as the illustration suggests. For example, RFCs and negotiations can be performed before, after or parallel to making the project ready for offshoring. Likewise, if the offshore provider is already known, then the project feasibility study leading to a final commitment to offshore the project can be done together with the stage in which the project is made ready for offshoring.

Furthermore, there are break points in the process which are not explicitly marked in the figure. Such a point, where the project may be canceled or the process may go back to an earlier stage, is, for example, the end of the offshore project feasibility analysis. If the offshore provider cannot credibly assure that the project is in good hands, the customer may negotiate with a different provider, or the offshoring idea may be completely canceled because the problems occurring with this provider are expected or presumed to be same with other providers.

The Offshore Provider's Perspective

Unless the offshore provider is a subsidiary or a captive center of a multinational enterprise, it is likely to be a normal software company in its home country, perhaps specialized in working with customers abroad. From the point of view of such a company, an offshoring project is just another project that has to be acquired on the market, bidding against competitors.

With regard to process models, offshore providers for their part are restricted by the overall model imposed by the customer. However, the further up in the system life cycle the offshore provider enters the process, the more freedom they have to form the process according to their needs and experience. Offshore providers have their own internal process models, even though they only expose the external interfaces of these models in the form of a global delivery model (GDM, e.g. [4]).

When the entire system development process is in the hands of the offshore provider, this organization is free to choose an iterative or evolutionary approach, for example. If the customer is ready for continuous engagement and collaboration, and if the offshore provider has onsite personnel, even a comprehensive iterative approach such as RUP may be applied.

At the beginning of the offshoring trend, maintenance as well as coding & testing projects constituted the majority of offshore projects, because programming work was inexpensive in the offshore countries. With accumulating offshoring experience, the level of software-capabilities maturity has grown over the years. As a consequence, many offshore providers were recognized as trusted partners for pre-coding stages as well as for outsourcing entire business functions or processes.

The more projects an offshore provider successfully completes with the same customer, the closer the business relationship with that customer becomes, and the likelier future contracts or awards in a bid are. With a long-standing relationship, trust between the partners grows, and the customer may be increasingly willing to outsource more critical process stages such as requirements analysis or even the entire solution process for the business problem to the offshore partner.

CONCLUSIONS AND RELATED PROBLEMS

Project management in offshore development projects is more complex than in onsite and onshore projects. Since project management activities are bound to process steps, adequate process models reflecting both onsite and offshore tasks are needed. This paper discussed common general process models as well as offshoring-specific models. A generalized offshoring life cycle model was presented.

The outsourcer's management face two major challenges: they need to communicate the benefits of offshoring for the company's competitiveness to their own staff and to manage the transformation process from inhouse to offshore development. In offshoring projects, there is still plenty of work left in the customer's organization. Yet this work is different, is more on the business and departmental level, e.g. preparing projects to make them ready for offshoring and identifying new opportunities for IS solutions. Personnel further down the development cycle may be qualified to complete tasks from within the project's life cycle, closer to the business problem, or in the coordination of onsite and offshore activities.

A final remark refers to the risks of offshoring. As organizations consider the vast benefits and allure of offshore outsourcing, they must balance the risks and uncertainties with the potential benefits. Many companies fail to manage the risks. A proper risk assessment and mitigation plan should be prepared in advance [13]. Infosys Technologies, for example, includes a detailed plan for risk identification, monitoring and mitigation as part of project planning. This plan covers risk identification, prioritization and mitigation options. The status of risks is continuously tracked and reviewed using a monthly milestone mechanism [7].

REFERENCES

1. *Agile Alliance: Manifesto for Agile Software Development*; http://www.agilemanifesto.org/ [accessed 19 March 2009].
2. Ambler, S.W.: *A Manager's Introduction to The Rational Unified Process (RUP),* Version Dec 4, 2005; http://www.ambysoft.com/downloads/managersIntroToRUP.pdf [accessed 20 March 2009].
3. Beck, K., Andres, C.: *Extreme Programming Explained: Embrace Change,* 2nd Edition; Addison-Wesley Professional, Boston, MA 2004.
4. Chaudhuri, D.: Off-shoring Process, Learnings and Case Studies; *Presentation at Euroforum Seminar "Offshore-IT-Projekte erfolgreich managen"*, Munich (Germany), Feb 17, 2004; online available at http://www.competence-site.de/offshore.nsf [accessed 12 Jan, 2009].
5. Cockburn, A.: Crystal Clear: *A Human-Powered Methodology for Small Teams*; Addison-Wesley Professional, Boston, MA 2004.
6. DC&M Partners LLC: *Offshore Development Process Model*; http://www.dcm-partners.com/offshore-development-process-model.htm [accessed 24 March 2009].
7. Infosys Technologies Ltd.: *Global Delivery Model – Risk Mitigation*; http://www.infosys.com/global-sourcing/global-delivery-model/risk-mitigation.asp [accessed 25 March 2009].
8. Kurbel, K., Pietsch, W.: A Cooperative Work Environment for Evolutionary Software Development; in: *Gibbs, S., Verrijin-Stuart, A.A. (Eds.), Multi-User Interfaces and Applications; Amsterdam et al*. 1990, pp. 115-127.
9. Lee, O.-K., Banerjee, P., Lim, K.H. et al.: Aligning IT Components to Achieve Agility in Globally Distributed System Development; *Communications of the ACM 49* (2006) 10, pp. 49-54.
10. Lewin, A.Y., Couto, V.: *Next Generation Offshoring – The Globalization of Innovation*; 2006 Survey Report; Offshoring Research Network, Duke University, Durham, N.C.; published in conjunction with Booz, Allen, Hamilton, Chicago, IL 2007; online available at: https://offshoring.fuqua.duke.edu/orn_report.pdf [accessed 24 March 2009].
11. Martin, R.C.: *Agile Software Development – Principles, Patterns, and Practices*; Prentice Hall, Upper Saddle River, NJ 2003.
12. Minich, M.: *Quality Assurance in Offshore Development Projects*; Master's Thesis, University of Applied Sciences Darmstadt, Germany; July 2007.
13. Morrison, P., Macia, M.: *Offshoring: All Your Questions Answered*; http://www.outsourcing-leadership.com/Offshoring.pdf [accessed 25 March 2009].
14. Ramesh, B., Cao, L., Mohan, K., Xu, P.: Can Distributed Software Development Be Agile?; *Communications of the ACM 49* (2006) 10, pp. 41-46.
15. Royce, W. W.: Managing the Development of Large Software Systems: Concepts and Techniques; IEEE WESCON Technical Papers, Western Electronic Show and Convention, Los Angeles, Aug. 25-28, 1970, pp. 1-9; reprinted in: Riddle, W.E. (Ed.), *Proceedings of the 9th International Conference on Software Engineering, Monterey, CA*; IEEE Computer Society Press, Los Alamitos, CA, March 1987, pp. 328-338.
16. T-Systems India: *Competencies, Service Offerings & Projects – Business Solutions: Offshore Delivery Model (Application Development)*; Company Presentation, Nov 2005.
17. Wells, D.: *Extreme Programming: A Gentle Introduction*; http://www.extremeprogramming.org/ [accessed 24 March 2009].