Bertrand Meyer
Mathai Joseph (Eds.)

LNCS 4716

Software Engineering Approaches for Offshore and Outsourced Development

First International Conference, SEAFOOD 2007
Zurich, Switzerland, February 2007
Revised Papers

Springer

Bertrand Meyer Mathai Joseph (Eds.)

Software Engineering Approaches for Offshore and Outsourced Development

First International Conference, SEAFOOD 2007
Zurich, Switzerland, February 5-6, 2007
Revised Papers

Springer

Volume Editors

Bertrand Meyer
ETH Zurich, Department of Computer Science
RZ Building, Clausiusstr. 59, 8092 Zurich, Switzerland
E-mail: bertrand.meyer@inf.ethz.ch

Mathai Joseph
Tata Consultancy Services
1 Mangaldas Road, Pune 411 001, India
E-mail: m.joseph@tcs.com

Library of Congress Control Number: 2007936130

CR Subject Classification (1998): D.2, K.6, K.4.2, J.1

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN	0302-9743
ISBN-10	3-540-75541-1 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-75541-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12171392 06/3180 5 4 3 2 1 0

Preface

SEAFOOD for Thought

Headline-grabbing though it may be, the software industry's large-scale allocation of work to developing countries has not so far generated much technical analysis. Attention is usually limited to the possible political and economic consequences, in particular the fears of loss of employment in the West. The aim of the present volume is different. We recognize that offshore development is here to stay, and not just a result of cost considerations. It is – more accurately – a form of distributed development, relying on advances in communications to let the software industry, in our globalized world, benefit from the wide distribution of human talent. But it is also the source of a new set of challenges, to which accepted software engineering principles and techniques have not completely prepared us. Producing high-quality software on time and within budget is hard enough when the QA team is across the aisle from the core developers, and the customers across the street; what then when the bulk of the development team is across an ocean or two?

The first SEAFOOD – Software Engineering Advances For Outsourced and Offshore Development – conference (prompted by an earlier article[1]) was an attempt not only to bring software engineering to outsourcing but also to bring outsourcing into the collective consciousness of the software engineering community. This is beneficial to both sides: successful outsourcing requires strong software engineering guidance, but research in the field must for its part account for the new world of software development. Whatever direction outsourcing takes in the coming years, we will never be just in one location any more.

SEAFOOD was held at ETH Zurich on 5–6 February 2007 and provided an opportunity for participants from academia and industry to confront experiences, ideas and proposals. The articles that follow are the result of this encounter. As can be expected of the first conference in such a novel field, we are still in the process of defining what constitutes a proper object of study on the topic; but the contributions already show a number of promising developments, which we are sure will be taken further in future conferences, starting from SEAFOOD 2008 to be held in the same venue in the first week of July 2008. The conference site at http://seafood.ethz.ch includes information on this conference, as well as past and future SEAFOOD events.

We hope that you will enjoy the results of SEAFOOD 2007 and that this volume will give you many useful ideas to understand and improve the engineering of outsourced software.

[1] Bertrand Meyer: Offshore Development: The Unspoken Revolution in Software Engineering, IEEE Computer, January 2006, pages 124, 122-123.

Many people contributed to making SEAFOOD 2007 a success. We are particularly grateful to the authors who submitted their work in a new and quickly evolving area; to the Program Committee members who reviewed the papers in time and through sometimes extensive discussions. (We extend our special wishes to Gio Wiederhold, who suffered an accident while in Zurich.) The role of Andrei Voronkov's excellent EasyChair conference system is gratefully acknowledged.

The conference benefited from four outstanding keynote presentations by Krishnamurti Ananthkrishnan, Chief Technology Officer of Tata Consultancy Services, Stuart Feldman, Vice President for computer science of IBM, Watts Humphrey from the Software Engineering Institute and Andrey Terekhov, from the State University of Saint Petersburg.

Martin Nordio from ETH played a key role in organizing the conference and helping prepare this volume; we are also grateful to Claudia Günthart for outstanding organizational support and to Christian Hunziker from ELCA for his work in publicizing the conference throughout Switzerland.

August 2008

Mathai Joseph
Bertrand Meyer

Organization

Program Co-chairs

Mathai Joseph, Tata Consultancy Services, India
Bertrand Meyer, ETH Zurich, Switzerland and Eiffel Software, California, USA

Program Committee

Manfred Broy, Technische Universität München, Germany
Kokichi Futatsugi, JAIST, Japan
Victor Gergel University of Nizhnyi-Novgorod, Russia
Koichi Kishida, SRA Key-Tech Lab, Japan
Qiaoyun Li, Motorola, USA
Mingshu Li, Chinese Academy of Sciences, China
Andrey Terekhov, State University of Saint Petersburg and TEPKOM, Russia
Gio Wiederhold, Stanford University, USA

Publicity Chair

Christian Hunziker, ELCA, Switzerland

Organizing Committee

Claudia Günthart, ETH Zurich, Switzerland
Martin Nordio, ETH Zurich, Switzerland

Table of Contents

Offshore Software Development: Transferring Research Findings into the Classroom

Kay Berkling[1], Michael Geisser[2], Tobias Hildenbrand[2], and Franz Rothlauf[2]

[1] Caribbean Artificial Intelligence Group CAIG, Polytechnic University of Puerto Rico, Electrical and Computer Engineering and Computer Science Department, 377 Ponce de Leon Ave, Hato Rey, PR 00918, Puerto Rico
kay@berkling.com
[2] Lehrstuhl für ABWL und Wirtschaftsinformatik, Universität Mannheim, D-68131 Mannheim, Germany
{geisser,hildenbrand,rothlauf}@uni-mannheim.de

Abstract. Distributed software projects are becoming increasingly commonplace in industry. Yet, software engineering education rarely graduates students with the necessary skills and hands-on experience that are particular to off-shore software development projects. Three key areas in successful off-shore software development projects are well documented in the literature as communication, knowledge management, as well as project and process management. This paper maps tasks within each of these three areas to functions that have to be provided by remote collaboration platforms and tools that distributed projects rely on. A case-study of an off-shore requirements engineering class experience between a Master course of Polytechnic University of Puerto Rico and a customer in a Swiss financial institution shows a correlation between areas of learning by the students and functionalities covered with the tools used in the classroom. The paper identifies additional tools, developed by the authors, which will provide additional functionalities in the deficient areas to increase the learning and preparation of the students for off-shore software development projects.

Keywords: Offshore Software Development, Distributed and Global Software Development, Software Engineering Education, Development Tools, Collaborative Software Development, Requirements Engineering, Traceability.

1 Introduction

1.1 The Fundamental Problem of Global and Offshore Software Development

Software development projects have never been easy to manage or predict in terms of cost, quality, or time to delivery. While a variety of methodologies exist to estimate cost and manage projects, still far more than half of all IT projects “fail” because of budget overruns, high maintenance costs or mismatch between desired and delivered functionality [31]. Such failures can in part be attributed to non-standard processes but are often due to inadequate communication between the parties involved [17].

In the last five years there has been a major increase in efforts to outsource software development to offshore locations such as India, China or Russia in order to cut the development costs [24]. According to Gartner, worldwide spending on offshore research and development will increase by a factor of 9 to ultimately $12 billion by 2010 and application development services will reach expenditures of $50 billion dollar [23]. On paper, the savings for projects that can be outsourced to economically advantageous locations look fabulous. The reality, however, is much less documented and shows that the challenges already faced by local projects are even enhanced by distance. The lack of effective communication due to distance, culture and language issues may well cause damage to projects that outweigh any potential savings of off-shoring development.

Computer science graduates who enter this global work environment are generally ill-prepared for these aspects of their future job. Few computer science curricula contain components to train their students in offshore development practices and related special considerations [10,27]. Yet, experience and skills in IT Management of global software development projects are essential to the success of offshore projects.

1.2 Objective and Methodology

The overall objective of this paper is twofold: First, the teaching methods used for offshore development education we present shall be replicable and reusable in various international university contexts. Second, by using methods and tools satisfying realistic offshore requirements, our basic approach will eventually be transferable and applicable in industry. Therefore, our work aims at making both a short-term and a long-term contribution to the improvement of offshore software development (OSD) practices: Through better education as well as methodological and tool support.

Analyses of the state-of-the-art in OSD practices, especially concerning software engineering (SE) methods and tools applied, yield three major categories of problems that need to be addressed – also in the classroom: communication, knowledge management, as well as project and process management [14]. The authors propose a combination of commonly used commercial and open source tools to provide the supporting functionality to improve performance of offshore projects in each of these three areas. Tools are chosen to be deployed in the classroom and support distributed educational software projects within semi-commercial settings, i.e. with a real customer. The classroom experience is designed to provide feedback with respect to accomplishing learning objectives and measuring the usefulness of the proposed supporting functionalities within real OSD scenarios.

The rest of this paper is organized as follows: Section 2 discusses current issues in OSD and presents major requirements for tools to support these scenarios. Section 3 describes the teaching environment for a SE course that is used as a baseline for this paper and outlines the deficiencies in this learning environment. Section 4 forms the theoretical component of this paper, where the necessary functionalities for effective OSD projects are evaluated and areas to be strengthened with additional tools are identified. In Section 5, the missing functionalities are mapped to new tools that are reviewed and their deployment in the classroom described in Section 6. Finally, the paper closes with a review and description of necessary future work to evaluate the success of the deployments.

2 Offshore Software Development: Issues and Requirements

Within the field of SE, literature on OSD heavily relies on the findings within the fields of distributed software development (DSD) and global software development (GSD) respectively. Besides issues of physical distribution, OSD also takes people's different mindsets and cultures into account when analyzing methodology and improvements in project management.

Herbsleb and Moitra [17] classify the problem classes most often encountered in GSD and OSD projects as follows: (a) strategic issues, (b) cultural issues, (c) inadequate communication, (d) knowledge management issues, (e) project and process management issues, and (f) technical issues (for complementary analyses see also [5, 15]).

Each one of these problem fields demands different approaches and tools. This paper will not be primarily concerned with strategic and cultural issues due to the fact that these problems may better be solved in business administration and social science respectively. Problem classes (c) to (e) however require tool supported solutions and are therefore our main focus – (f) in our opinion characterizes a cross-cutting concern and is hence not analyzed separately. Each of the three areas described below overlap somewhat. For example there is no project management without communication and visualization of meta-data is important for all three categories. However, the categories can be broadly separated as follows: Project and process management issues correspond to coordination problems: e.g. synchronization and mutual awareness in concurrent globally distributed processes [17]. Communication issues pertain to a broader range of SE tasks, whereas knowledge management is the most abstract problem class as regards the scope of activities affected.

2.1 Project and Process Management Issues

Process management is highly critical in distributed scenarios. Software process coordination is mostly about the division of labor between distributed sites and developers: Tasks can either be divided according to the code structure or different development disciplines [29]. Either way, parallel concurrent processes have to be managed carefully [17], while still guaranteeing process flexibility and integration of different methods from the various sites [18].

As Requirements Engineering (RE) is the most critical phase in OSD [25], a systematic proceeding will be needed to provide efficient client integration and decision support for requirements selection even though physical meetings might not always be possible [12].

Especially in OSD, roles are highly important to help the coordination of a large number of developers [20]. A team member can take on one or more roles within a single project and consequently can be a developer and later a tester, with duties varying accordingly.

Empirical studies suggest that informal communication is a very important aspect of coordinating teams in uncertain tasks such as software development [6, 20]. The physical distance between sites makes it harder for distributed team members to spontaneously and informally communicate with other team members in order to coordinate their work [14]. This limitation within OSD implies fewer coordinating interactions since developers find it more difficult to discern people's current activity and whether it is appropriate to interrupt them at a certain time [1]. It can also mean that such developers encounter greater difficulties in coordinating OSD projects as a result. Therefore, team awareness and process transparency are crucial for OSD.

Moreover, change management and impact analysis are particularly critical coordination tasks in OSD. Distributed developers with different processes and tools make it even more difficult to coordinate changes to the code base and prevent conflicts [24]. Impact analysis, i.e. seeing the consequences of your changes in advance, is also significantly harder in distributed settings such as OSD projects since related artifacts are also most likely distributed over multiple sites [1].

Visualization and understanding of complex contexts, e.g. processes or artifact dependencies, greatly support the project and process management by providing a better view of the information.

2.2 Communication Issues

Software development is a very communication-intensive activity and issues raised of an inadequate communication are even more complex in OSD [17]. As had been mentioned before, distance introduces barriers to informal communication which leads not only to coordination issues [5, 14]. It also makes it difficult to establish trust and form relationships among distributed stakeholders [8].

Comparable to the problem of process integration, the integration of different communication tools – synchronous and asynchronous – must also be seen as a major issue in OSD. These tools need a well-defined interface because fuzzy interfaces will mostly lead to inefficiencies and other technical problems [17].

Moreover, visualization and understanding of complex contexts, e.g. processes, artifact dependencies, and social networks of developers, are often problematic as well. However, these visualizations are critical to support formal, project-related communication in order to make it more efficient. This issue is even more critical in OSD arrangements since visualizations can help overcome language barriers [11].

2.3 Knowledge Management Issues

In addition to coordination and communication issues, general knowledge management is vital to fully exploit the OSD potential [17]. In distributed projects, the physical location of information artifacts such as source code, task descriptions, or comments on changes, and the lack of "global knowledge" about their existence make traceability and rationale management an especially hard task in OSD [7].

Moreover, poor knowledge management leads to many missed reuse opportunities that otherwise would have potentially saved lots of time and money [17]. Global knowledge management (through visualization) is also critical in order to determine the overall status of the project at any given time, e.g. critical paths of activity flows and buffers among subsequent tasks [17].

In GSD, in addition to documenting the various artifacts, updating and revising the documentation is especially important since team-members are not all collocated. The usage of visualization tools is only as useful as the information within the tools is correctly updated. Automating such updates becomes particularly important. To prevent assumptions and ambiguity and to support maintainability, documentation must be current and reflect what various teams are using and working on [17].

3 Offshore Software Development in the Classroom

At Polytechnic University of Puerto Rico a recently established track in SE focuses on student learning in OSD. The students participate in a series of related classes including Software Engineering I (SE1: Foundations in Software Engineering Management and Methodology), Software Engineering II (SE2: Requirements Management with offshore component), and Software Engineering III (SE3: Offshore Software Development). In all cases, the emphasis is placed both on the project management and development process as well as the effective use of supportive technology and less on the quantity of project that was completed. This section describes the course setup followed by the learning objectives that are aligned with the issues identified in Section 2.

3.1 Classroom Scenarios

For the purpose of defining three types of classroom OSD scenarios, the division of labor splits according to SE workflows based on the Rational Unified Process (RUP) [21] into three tiers: Client (Business Idea), Intermediary (Requirements Specification, Analysis, Design, Implementation at the prototype level, Configuration and Change Management, Project Management), and Supplier (Implementation, Environment, Testing). Client, Intermediary and Supplier are each situated in a different location and time zone. The Client, in our case, is a single person from industry who has a project idea and has to agree with the final product. The Intermediary forms a buffer between client and supplier. This buffer has the function of alleviating the work load on the client side, dampen cultural differences and provide quality insurance. For the purpose of the classroom, the Intermediary takes on two functions in separate semesters.

In SE2, the class takes on the task of analyzing and specifying the requirements of the client in detail and building a prototype. In the second semester, the Intermediary is responsible for defining the technology, designing the software architecture, and overseeing the supplier class in building the entire project according to specifications. The Supplier provides the implementation according to specifications of the Intermediary. As a result of these high-level roles, two relationships can be emulated in the classroom: between Client and Intermediary, and between Intermediary and Supplier according to Figure 1. The split according to SE disciplines is described as follows for SE2 and SE3:

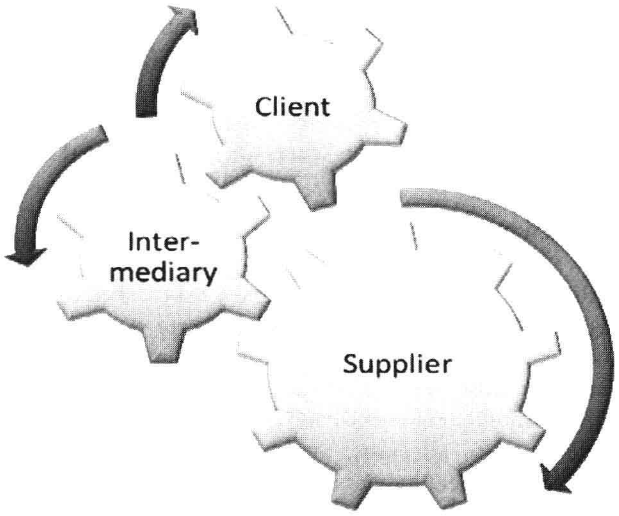

Fig. 1. Class roles include Client, Intermediary and Supplier. In SE1, these roles are collocated within the same classroom to teach the fundamentals of SE projects. In SE2 (Offshore Requirements Engineering) Client and Intermediary are located in different time zones. In SE3 (Offshore Implementation) Intermediary and Supplier are residing in different locations.

a) Requirements Engineering (between Client and Intermediary) in SE2

A client in a distant location poses the project idea in form of a short description of one to two pages. Usually an official or unofficial industry partner plays the role of a client. A typical profile of a client can be:

- Someone who does not have the resources to define and follow up on a project but has a strong interest in seeing it developed further in order to have a detailed specification and a prototype.
- A client who is training to become a manager for a global team and is supported by the company in participating with the class as part of a training.
  The class defines the project but additionally has to ensure that the client agrees with the specifications.

b) Implementation (between Intermediary and Supplier) in SE3

The Intermediary and Supplier class pair takes on a project resulting from collaboration between Client and Intermediary in order to specify the technology in more detail. The Intermediary supervises the development in accordance with the Client, where now all three (Client, Intermediary and Supplier) are situated in different locations – but not all in different time zones. E.g. Puerto Rico and Chile are in the same time and language zone which is exactly what makes this setup so attractive. The supplier codes the project assignment according to the specification of the intermediary and interacts with the intermediary only. A variation on this scenario is a role reversal between the participating classes in order to emphasize understanding of the entire process.