Juho Mäkiö, Stefanie Betz, Rafael Prikladnicki

Distributed Software Development

Methods and Tools for Risk Management

ICGSE Workshop 2008

Proceedings

Proceedings

Distributed Software Development

Methods and Tools for Risk Management

ICGSE Workshop 2008

17. August 2008

Bangalore, India

Table of Contents

Distributed Software Development
Methods and Tools for Risk Management

ICGSE Workshop 2008

Call for Papers

Distributed Software Development – Methods and Tools for Risk Management

August 17, 2008 ***http://icgse2008.di.uniba.it/***

Bangalore, India

Co-located with ICGSE 2008 Bangalore, India, August 17-20, 2008

Important dates

June 30, 2008 (updated)	Decision of acceptance to paper authors
July 13, 2008	Final version of accepted papers due, according to IEEE standards
August 17, 2008	DSDMTRM Workshop

Organizers

Dr. Juho Mäkiö, FZI Research Centre for Information Technologies (maekioe@fzi.de)
Stefanie Betz, University of Karlsruhe
Rafael Prikladnicki, PUCRS, Brazil

Invited Talk

Dr. Ita Richardson: Globalising Risk Management in Safety Critical Industries
Lero – the Irish Software Engineering Research Centre and University of Limerick
Ireland (http://www.lero.ie/)

For software companies to thrive, they must avail of business opportunities that present themselves. Such opportunities include the development of software for safety-critical industries such as the automotive and the medical device industry. In both of these industries, there has been a significant growth in the amount of software in the finished product. However, when software is developed for these industries, cognisance must be taken of the safety critical standards required by organisations such as the International Standards Organisation (ISO) and Food and Drug Administration (FDA) for marketing purposes. Combining these standards with software development standards such as Capability Maturity Model Integrated and ISO15504 can cause difficulty for the software developer. A further issue to be catered for is distributed software development, particularly when it is globalised. Add this issue to the complications imposed by safety-critical standards and software standards – ultimately software project managers have a conglomeration of issues to deal with.

In this talk, Dr. Richardson will look at the implementation of software Risk Management for safety-critical industries, focusing on medical device software development which takes account of both software development and safety-critical standards. She will then examine some of the issues which must be catered for implementing successful global software development and suggest how safety-critical risk management can be implemented in such projects.

Technical description

Motivation

The strategic reasons for a company to opt for distributed software development are among others to speed up the time-to-market, to access the global resource pools, to profit from the around-the-clock development, and to reduce the costs. The distribution may be organized in multiple ways: The Company may cooperate with an onshore or offshore provider or the distribution is organized within the company by captive outsourcing. The distribution of software development projects has become a key software development technique. But, the distribution is a challenge. For example, various studies report about failures in outsourced software development projects. Hence, there seems to be a gap between the expectations and the reality regarding the results. In reality, the actual savings may not fulfil the expectation, or even, in worst case, no cost savings at all were realized. This is caused by multiple reasons like the lack of methodological foundations, insufficient least standardized models, ignorance of the risks of offshore software development projects and increasing complexity of software development projects. Do we have any practicable methods to select suitable projects from a set of possible projects? How does project portfolio management need to be organized for offshoring?

Various aspects of distributed software development are discussed in the current literature. However, there are still several gaps in planning and accomplishment of distributed software development projects. Reasons for negative incidents are, for example, poor project management and missing skill management. Thus, suitable tools and methods taking into account the specific needs of offshore software development are required. What are the specific needs? How and by which tools these needs may be fulfilled?

Issues of risk management are highly relevant, when an organization distributes the software development project. Fitting risk management tools and methodologies are required as the ignorance of the risks may lead to various undesirable and costly events. But, how should one manage the risks? What tools and methodologies are required? What is “risky” in the distributed software development and how the risks may be avoided?

Successful organization of geographically distributed teams requires additional efforts. Different approaches should be taken for dividing and allocating tasks. Additional infrastructure and tools need to be implemented. Time and communication delays are often inevitable in an offshore environment. It is therefore more difficult for the whole team to react to emerging events or changes. How can tool support the communication during the project? Which information needs to be communicated to whom and when?

Furthermore globally distributed software development projects also raise new challenges for the education of future software professionals. In addition to technical skills, there is obviously a need to focus on the education of high-level system skills, project management issues, and analytical as well as synthetic reasoning techniques in the software engineering field.

Technical issues

The challenges that are coupled with the management of distributed software development projects call for new process models, techniques, methodologies and tools. This workshop focuses on multiple aspects of management of distributed software development.

Topics of interest to the conference include the following aspects of global software engineering projects (but are not restricted to):

- risk management / risk modelling / risk analysis of offshore projects
- process for planning and execution
- organizational models and strategies
- project portfolio management
- communication models for distributed software development teams
- causalities between critical success factors
- metrics for critical success factors
- problem handling during project work
- outsourcing / offshoring contracts
- decision support systems
- strategic planning
- project planning and preparations
- simulation

Goals

The challenging issues arising in the field of offshore software engineering projects require novel approaches in risk analysis, project planning, and methods in order to handle the bounded financial and technical risks.
The goal of this workshop is to provide a forum for researchers and professionals interested in global software development to discuss and exchange ideas. In particular, this workshop takes the perspective of the practitioner and focuses on the techniques that will help software professionals to meet the unique challenges in a global development environment. Thus, the major goal of this workshop is to discuss novel methodologies for risk management for global software development. Additionally, we want to provide a platform bringing together researches and practitioners in order to share their knowledge and requirements in the field of offshore software development.

Target Audience

- software engineers
- computer scientists
- business process engineers
- project managers

Program committee

- Jens Borchers, steria mummert consulting
- Prof. Dr. M. Esser, St. Petersburg State Polytechnic University
- PhD. Robert Feld, Blekinge Inst. of Technology
- Prof. Dr. Andreas Gadatsch, University of Applied Science Bonn-Rhein Sieg
- Prof. Dr. Eila Järvenpää, Helsinki University of Technology
- Dr. Andreas Kotulla
- Prof. Dr. Andreas Oberweis, University of Karlsruhe
- Dr. Maria Paasivaara, Helsinki University of Technology
- Dr. Darja Smite, Riga Information Technology Institute
- Rolf Stephan, Centre for International Collaborative Software Development
- Prof. Dr. Walter Tichy, University of Karlsruhe
- Dr. M. Wiener, Friedrich-Alexander University Erlangen-Nuernberg

Paper submission

- Papers must be submitted electronically by email to the organizers (maekioe@fzi.de)
- Your paper must conform to the IEEE proceedings publication format (8.5" x 11", Two-Column Format) described at IEEE/CPS
- You paper should not be longer than 6 pages including all text, references, appendices, and figures
- Your submissions should be in PDF format
- Submissions that exceed the page limit (6) or do not comply with the proceedings format will be desk rejected without review
- The results described must not be under consideration for publication elsewhere

Accepted papers will be published online and as workshop proceedings.

Globalising Risk Management in Safety Critical Industries

Dr. Ita Richardson
Lero – the Irish Software Engineering Research Centre
and University of Limerick
Ireland

For software companies to thrive, they must avail of business opportunities that present themselves. Such opportunities include the development of software for safety-critical industries such as the automotive and the medical device industry. In both of these industries, there has been a significant growth in the amount of software in the finished product. However, when software is developed for these industries, cognisance must be taken of the safety critical standards required by organisations such as the International Standards Organisation (ISO) and Food and Drug Administration (FDA) for marketing purposes. Combining these standards with software development standards such as Capability Maturity Model Integrated and ISO15504 can cause difficulty for the software developer. A further issue to be catered for is distributed software development, particularly when it is globalised. Add this issue to the complications imposed by safety-critical standards and software standards – ultimately software project managers have a conglomeration of issues to deal with.

In this talk, Dr. Richardson will look at the implementation of software Risk Management for safety-critical industries, focusing on medical device software development which takes account of both software development and safety-critical standards. She will then examine some of the issues which must be catered for implementing successful global software development and suggest how safety-critical risk management can be implemented in such projects.

Challenges in Execution of Outsourcing Contracts

Nagesh Mukunda Rao,
Logica private limited, Bangalore
e-mail: Nagesh.rao@logica.com

Abstract

In today's world, companies try to maximize outsourcing to stay competitive in the market. The companies worldwide acknowledge the fact that to stay ahead in competition; they need to reduce costs, deliver the best quality, use latest high-tech skills, and be reliable and creative.

In the software industry approximately 50-70% of the revenue generation with major players comes from outsourcing deals. In many cases these deals are moved from one organization to another due to mismanagement by the service providers, leading to unsatisfied customers.

This paper provides the insights in to the list of common issues faced during the execution of the outsourcing deals. The paper is more focused on problems faced by small to medium sized organizations in handling the outsourcing deals and the approaches implemented towards resolving these issues.

1 Introduction

Most of the outsourcing contracts range from of 5-10 years. The major focus of these contracts is cost savings. Customers are trying to find out long term solutions with short term objectives keeping the cost in mind. According to Gartner report "Four out of five outsourcing deals will be renegotiated during the lifetime of the contract, because many deals have been too focused on cutting costs."[3]

In Figure-1 from service provider prospective during initial phase, the focus is on the knowledge acquisition and delivery. During this period the service provider may incur losses due to learning curve involved in understanding the applications, building the relationship with the customer, overseas travel and trainings.

During the maturity phase there is lot of focus on implementing the learning, processes optimization, understanding the customer expectations, the service provider slowly starts making profit. The recovery period may vary from six months to one year based on the type and size of the contract.

In Figure-1 from the customers prospective during the initial phase of the engagement there may be increase in expenses due to maintenance of their IT team along with the expenses incurred to build the capability with the service providers, establishing infrastructure (HW/SW/link), knowledge transition, recession and many more.

During maturity phase after the service provider takes up the activities, there shall be jump in revenues due to the reduction in operating costs. During maturity phase, there is lot of opportunity to streamline operations.

During the optimization phase, the operations are more or less stable and there will be less opportunity for improvement. This will continue till the end of the contract unless there are innovative ways to bring down the costs.

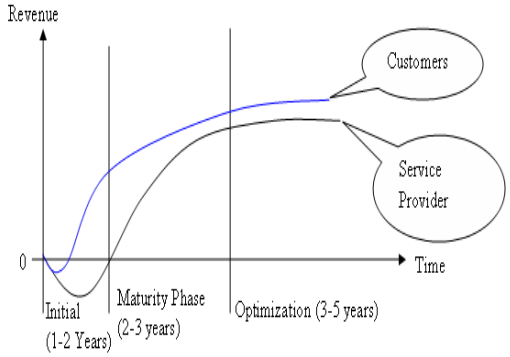

Figure-1. Service provider and customer operation performance

2 Key Challenges

Following are a few key challenges faced during the execution of outsourcing contract.

2.1 Contract Management

This is the key for the success of any outsourcing deals. The contracts are written by the central contract team and the execution is done by a different team who don't have the in depth understanding of the contract. The contract is considered highly sensitive and confidential. From the author's prospective, not all parts of the contract are confidential and sensitive. As a part of the mandatory training include contract awareness sessions.

Have a dedicated contract manager in the account to manage the changes to the contract. As a part of the process include contract manager's approval before starting any new piece of work to make sure the identified piece of work falls within the boundaries of the contract.

Most of the time the assumption is that the contract can't be changed after it has been agreed and signed. It may be true for short term contracts (up to one year), but this doesn't hold good for long term contracts (> three years). It takes six months to one year to realize the assumptions made at the time of signing the contract. The long term contracts should have provision to review and renew at least once a year, due to the following reasons.

The size and complexity of the application changes over the period of time, this should be reassessed and agreed again among the parties. For e.g. in one of the application the development team has added more than 10,000 lines of code in a year's time, changing the complexity of the application from medium to high. The product road map indicated major enhancements to the application over the next two years. This will have an impact on the team size required to support the application.

In another contract, it took three years to realize that the contract was based on 7 hours work day (Actually the organization works on 8 hours per day). Due to this the service provider lost ((50*1)*(250*3)*0.7)= 26250 hours or 3750 days (with 7 hours of work day) of billing in three years (The calculation is based on with a average team size of 50 people, organization work day is 8 hours and the contract was pointing at 7 hours work day, there is a loss of one hour a day, 250 working days in a year, 3 years duration, @ 70% utilization). At the rate of USD 200 per day over the period of 3 years the performing organization had lost USD 750,000.

In another case, the hardware and software upgrades were considered as a part of the normal support. It has been agreed in the contract to keep the entire hardware and software up to date (one version below what is available in market). When the analysis was done, it was realized that all SW and HW are at least three versions behind what is available in the market and required at least three upgrades, meaning the work has increased by three folds.

Another common mistake in the contract is, year after year, some percentage of savings is committed by optimizing the operations. For the first three to four years there is scope for optimization and beyond this period there is very little or no scope for optimization. The performing organization should be careful about committing the number.

In another case, the customer had committed 15000 days of development effort each year. Unfortunately, customer only provided 60% of committed work and maintaining the minimum team size ended up in making a loss. The contract was amended a year later.

The contract may call for the offshore teams to support during customers working hours (in a different time zone). The working hours has a direct impact on the employee's social life leading to low morale. This in turn affects productivity. Not all teams need to work during the unsocial hours; the development team may continue to work in the normal hours, with 2-3 hours of overlapping time with the customer. The support team members can work from home for supporting the applications during unsocial hours. These points need to be agreed with the customer in advance.

2.2 Demand Supply (Pipeline) Management

The application support team operates on number of resources required to support the application and the billing remains constant. The invoice generated by the development team depends on the actual effort spent in the projects. To achieve considerable margins from the projects, it is important to maintain at least 85% billable utilization. For this to happen, agree with the customer on the pipeline at the beginning of every year. This exercise should be completed within the first month of the year. It is important to morally commit to this plan by both the parties. This is one of the critical characteristics of the matured program.

The biggest challenge in today's Indian IT industry is to get the right skilled people within the specified time frame. It takes a minimum of six to eight weeks to get the resource on board and if it is a niche skill the lead time may go beyond 90 days. With this scenario, it is very difficult to arrest the sporadic demands.

Following are the list of approaches used in flattening the demand and making it more predictable.

- At the beginning of the year, the service provider and the customer need to agree on the list of projects with the tentative start and end dates with month wise effort forecast (Enterprise plan). If the demand peaks during certain months, agree with

the customer to push or pull the project by few days to smoothen the demand.

It is recommended to use Microsoft Enterprise Project Management System (EPMS) tool to forecast the resources by loading all the (current and future) project plans on to the tool. The tool can provide weekly resource loading details. Discuss this report with customer in the monthly enterprise planning meetings.

Figure-2 represents the enterprise plan. The initial plan had a fall during November to February, peaking between July and October. After reprioritizing workshop with the customer the demand has been smoothened out.

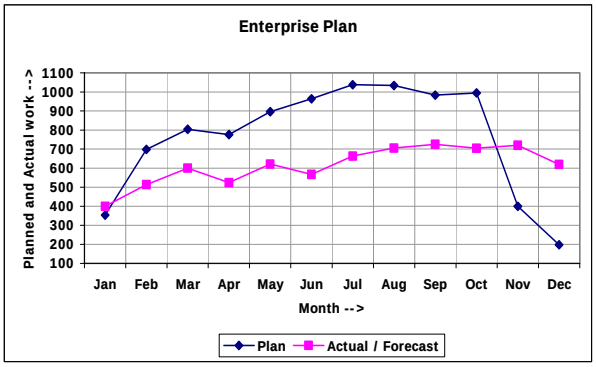

Figure-2 – Enterprise plan

- Have the core team defined consisting of Business analyst, Technical architects and technical leads who have the most knowledge of the business, technology and application. The core team contributes to about 30% of the team strength. The core team prepares the technical specification to the level of pseudo code and the resources joining the team are expected to start coding based on the technical specifications. Additionally to improve the margin, most of the organizations are mandating to use 50 % of low skilled resource (less than two years of experience) and remaining high skilled resources in the team.
- Have a common pool of people shared across two to three programs.
- Use of support resources in development projects. In the author's experience, it is possible to use up to 30% time of the support staff in development projects. This may need to be agreed with the customer based on the type of engagement. This improves the utilization and keeps the employee morale high.

Resource optimization is a two stage process, first optimize the team utilization individually, then merge the teams and apply second level of optimization. This will provide the best results.

2.3 Process Definition refinement

Since the customers core business is in different domain (like insurance, finance or manufacturing) they may give least importance to IT processes. In the authors experience the processes used by the customers are not matured enough. The common problems that may arise due to lack of process definitions are

- The process may not address the agreed Key Performance Indicators (KPI's)/Metrics
- The program will be left out from the organization process upgrades, new standards and certifications.
- Since the processes defined by customer are not standard, the user's interpretation may be different from what is intended.
- There may not be enough support from the central quality team as the process deviates more from performing organization standard processes.
- Lack of training available on the use and application of the customer defined processes
- The scope for the process improvement will diminish

Most of the IT organizations are ISO 9001 tick IT and/or CMMI level-5 certified. It is recommended to follow steps mentioned below in defining the processes while defining and agreeing the process

- Understand the customer process and involve the central quality function in discussions with the customer
- Define and agree the key performance indicators and quality goals.
- List and understand the gaps in the customer process from the organization process standards.
- Modify the process to suit the customer needs and make sure these are inline with organizational process standards. While defining the process make sure the following points are taken care of[4]
  - Remove unwanted actions / waste / over heads
  - Enable capturing of defects in early phases of the project
  - Design the process in such a way that it falls through the set of defined actions and make sure to close the feedback loop to arrest any gaps.
  - Have right number of check points/ quality gates defined in a process. If the checkpoints defined are less than the required, then the defects seep through to the next phase of the project. If there are more than required this becomes a process over head.
  - Plan for a group review of the deliverables wherever possible and is more effective
  - Achieve consistency in delivery
- Within the first three months discuss and agree the processes with the customers

- Train the teams and rollout the processes from day one.
- Encourage the teams to use process tailoring guidelines to suit the project needs [1].
- Conduct timely audits and interviews with the process users to identify the gaps and the improvement areas.

2.4 Tooling

The performing organizations are failing to realize that they are the producer of the world class SW applications and fail to spend less than 3-5% of the effort a year towards the development and upgrading the tool. The engagement should start with the identification of the minimum set of tools that helps the teams, management and customer in providing consistent, reliable and accurate data. From the author's experience, the two difficult problems in any outsourcing deals are

- Addressing an invoicing challenge (Invoicing challenges reflect very badly on the credibility of the performing organization) and
- Trying to find an answer for the challenges raised on the numbers reported in the reviews with the customer

The following are the three minimum tools that are required for the smooth running of the program

2.4.1 Time sheet management system - This is the basic system to begin with any outsourcing contracts. All the invoicing happens through the inputs provided by the time sheet; it is important to automate the time sheet management system as a first step to improve the accuracy of the invoicing and reduces the invoicing challenges drastically.

Following are the list of problems with manual time sheet management.

- Use of wrong or miss spelt project code.
  - Project Manager or service manager doesn't realize until customer raises it as invoicing challenge.
  - Modifications of the past time sheet data leads to invoice adjustments to the future billing cycles and may lead to invoice challenges.
- Missing time sheets, this is straight away categorized as revenue loss.
- More time spent by the managers in fixing the errors and providing the clarifications for the challenges.
- Inconsistency in reports

In the author's experience the invoicing challenges can be reduced (by more than 95%) by following simple time sheet disciplines.

2.4.2 Service Management - Usually customer provides the tool or there may be a standard tool (e.g. remedy) used at the organization level by the service provider. In case if both options are not available, develop a simple tool to capture the call, status of the call, description of the action taken with the date, time stamp and resource name. The system should report the call status against the key performance indicators like response time, resolution time, work around time, available time and also act as a knowledge base.

2.4.3 Project management system – In medium sized contracts, at any given point of time there will be at least 40-50 work requests (projects) under progress. It is a difficult task to manually track them. What is required is a simple project tracking tool which can maintain one line entry for each work request (WR) providing the details like WR number, description, PM name, estimate, approved date, planned and actual dates and efforts. The tool should be able to generate simple reports on project performance against the agreed key performance indicators.

Use of Microsoft enterprise project management (EPMS) tool is found very useful in managing the projects in the account, forecast and resources. Customers are provided the access to the EPMS reports and project dash boards (There were reports customized in EPMS to suit the account/customer needs).

Use of test management tool like test director was very useful in managing the test cases, resource assignment and monitoring the progress. The workflow built in the tool helped the team in managing the defects.

On-line review of the project with management and customer is happening through these online systems, providing the most recent status of the project. This was very well received by the customer.

There is no limit to automation; the above are a few basic tools an account should start with. The second phase is to integrate these tools and the third phase is to enhance them to suit the account growing needs.

2.5 Culture and Communication Management

The offshore team sits thousands of miles away in a different time zone. The tendency is to overlook at the cultural issues, communication styles, difficulties in understanding each other's context for making decisions and differing traditions regarding codes of conduct related to authority and work procedures are common[3].

Following are a couple of critical issues popping up due to cultural mismatch

- In some cultures (such as India), questioning authority is considered disrespectful. As a result, offshore resources may be inclined to work exactly to specifications, even if they believe a more-efficient method can be used.
- Use of first, middle and last names appropriately in mails and telephone conversations - Westerners are very sensitive to this.
- Be clean and tidy. Dress appropriately as you would be representing your company and in some cases your country. (E.g. The Europeans are more particular about dress code. They expect the team to come in full suit.)
- They are sensitive to odors – So be careful while using body odors, scented hair oils, etc.
- Avoid having lunch at desk, specifically if the food is cooked using spices.
- In Indian culture saying "no" is considered disrespectful. The team should be trained to say "no" when required.
- Use of proper body languages like nodding the head, maintaining appropriate physical distance while speaking, establishing the eye contact etc.
- Avoid grouping and speaking loudly in local languages at customer locations
- If a Japanese or British national says "no or don't" it means a lot. They express agreements and disagreements very politely.
- Be sensitive to the cultural issues right from their holidays to their sports/games, political and religious beliefs. For example, knowing a few names of national foot ball team and their capability will help to build good relation with the customer.
- Don't work your weekends unless it is absolutely required. Go out and enjoy the local place, culture, tradition and attractions.
- Agree on the joint exchange program, in which customer work from offshore location once in a quarter for a week's time and vice versa. This will help both customer and offshore team appreciate their difficulties.

The above mentioned cultural issues may sound inconsequential but are serious and can be addressed through proper training. Provide focused, country and location specific cultural trainings. It will be advantageous to include both employee and the customer in the same training.

2.6 Employee Morale

Employee morale is an important yet often neglected area in organizations.

At one point in time there were 4 projects which had problems like schedule and effort over run (up to 400%), too many requirement changes, many delivered defects and many repeated incidents resulting in customer escalations. The employee morale dropped drastically as the number of customer escalations increased. The best contributors are stressed more, creating an imbalance at work. The frustration spread quickly throughout the account and no one would come forward to work for the account. The employee's gets a great relief when they are released from the account.

During high-stress periods of heavy overtime / low morale, the following recommended steps were implemented which helped the team to convert the pain in to power.

- Take a stock of the situation and agree with the customer on the new delivery dates.
- Co-locate the team. That gives them the feeling they are not alone in the journey.
- The high level deliverables were broken down to daily deliverables and tracked closely. The delay's and road blockers were arrested at the earliest with a clear fallback plan.
- Clearly define the threshold limits for escalation, no one is allowed to spend more than two hours on any issue. The problem should be immediately escalated to next authorities and the respective team member should take up the next task. The project manager will identify suitable resource to resolve the issue [2].
- Take each opportunity to mix and talk to the team, for example join the team for lunch, take an opportunity to pick-up or drop your team member on your way to office or back home. This will enable informal communication channel and brings closeness with the team [4].
- Before going home, identify who are all planned to extend the work time. Organize transportation and dinner for the resources. This makes them feel that they have been taken care.
- Having the highest level leaders in the organization visit the workplace and recognize employees by name, expressing gratitude for their hard work.
- Create more opportunities for the team by providing the flexibility to move them across the projects of their interest and by involving the team in innovation (which has been detailed in section 2.7)

By following the above simple steps all 4 projects were delivered on the agreed timelines and the

employee morale improved drastically from 2.3 to 4.1 on a scale of 1 to 5 (1 being lowest and 5 being the highest). The same steps are being practiced across the account.

2.7 Innovation at Work

Innovation doesn't always mean product innovation. It's about business, business model, operational process, time-to-market or service innovation. Innovation need not necessarily be fresh ideas; it can be new way of doing things. It is about knowing and understanding the problem and be ready with the solution before the customer realizes the problem [5].

The ideas of innovations always go unnoticed as the team would keep working on the applications. Over the period of time, they acquire good knowledge of the application and business processes. It is just required to capture those ideas and progress that to innovation. There are three types of innovations.

Innovation which helps the service provider in optimizing the operations - This aims in improving and optimizing operations to meet customer demands. This also means more efficient ways to manage existing operations by improved knowledge management, communication, application monitoring etc.

Innovation that helps solving customer problem – This will help in winning customer confidence by showing them that service provider is really concerned about their challenge/focus areas. This makes the customer feel that applications are secure in service provider hands.

Innovation that helps business and service provider venture on new opportunities - Service provider who is closely involved in maintaining and enhancing applications can suggest the customer about the redundancy of the functionalities, opportunity to promote applications as products, cost effectiveness of integrating applications etc.

The steps involved in initiating innovation are as follows

- Kick off the innovation surveys at least twice a year, keep the survey free flow.
- Encourage the teams to identify the problems (problem identified is as good as finding 50% solution to it), to contribute new ideas.
- Collect ideas and segregate them in to process related, Problem, new business, etc.
- Identify the panel to analyze these contributions, the panel can discuss with the respective individual for clarifications.
- Select the top five ideas and help the team to prepare the business case.
- Find out whether this idea can be patented (There are few organizations, >20% of their revenue generation is through patents).
- Present the business case to the management; propose the ideas to the customer.
- Reward the contributors.
- Define a contact point in the program so that the team need not wait for the launch of next survey for contributing their new ideas.
- The other ways of innovations are at the individual team level brain storming, inviting teams to present papers on the new ideas etc.

3 Conclusion

The simple approaches listed in this paper has provided significant benefit in bringing the program under control and turned the loss making program in to a profitable one. The team morale has improved and employee satisfaction survey showed drastic improvements. The team had delivered more than twenty projects with zero defects during acceptance testing and no invoicing challenges in last six months. The customer escalations have come down drastically and the customer satisfaction rating has been improved to four on a scale of 1 to 5 in a span of a year. The program has become more stable, predictable and a reference site for other customers.

4 References

[1] Roger S. Pressman, Software engineering – A practitioner's approach, Mc Grew Hill, USA, 2005

[2] Dr. APJ Kalam with Arun Tiwari, Wings of fire, 2005

[3] Helen Huntley, "Five Reasons Why Offshore Deals Fail", Gartner report dated 31-May-05 ID number: G00127840

[4] Nagesh M, "Cycle Time Reduction", APSEC-06 13th Asia Pacific Software Engineering Conference, Bangalore, Dec 6-8, 2006

[5] Tony Davila, Marc J. Epstein, Robert Shelton, Making Innovation Work: How to Manage It, Measure It, and Profit from It, 2006

[6] Marcus Buckingham, Curt Coffman, "First break all the rules"

Unicase – an Ecosystem for Unified Software Engineering Research Tools

Bernd Bruegge, Oliver Creighton, Jonas Helming, Maximilian Kögel
Siemens Corporate Technology & Technische Universität München
icgse08@creighton.de, {bruegge, helming, koegel}@in.tum.de

Abstract

Many research approaches aiming at control and mitigation of risks in global software development (GSD) are based on tool support. Following a rigorous research approach these tools need to be evaluated and therefore implemented. Existing tools lack support for research requirements. As a consequence researchers often have to build their own solution from scratch. This is a time consuming task associated with the risk of failure. This paper proposes a platform called unicase. The goal of unicase is to support researchers in building tools for evaluation of research approaches in GSD. Unicase is based on Eclipse technology and helps the researcher in the evaluation of tool-supported approaches to GSD including approaches to risk management.

1. Introduction

A lot of research in Global Software Engineering (GSD) has been carried out in tools supporting GSD projects to control the additional risk associated with distributed project set-ups. The scope of these tools is often visualization (e.g. [10], [11], [15], [16]) and collaborative modification (e.g. [9], [10], [14]) of a custom set of artifacts. In typical software engineering projects these artifacts are stored separately in various repositories including databases, file systems or software configuration management systems. This includes tools for requirements engineering, UML tools for the design, project management tools, developer tools and many more. In this paper we focus on two core issues in using commercial tools for research in GSD projects.

(1) Limited domain model

Innovative tool support for GSD is often based on the introduction of new element types into a domain (e.g. [10], [14]). Commercial tools are limited to a predefined model from their domain and usually not geared towards extensibility. Furthermore Traceability between specific artifacts from different domains, e.g. between risks and system models [6], is a precondition for a variety of research approaches in GSD. A lot of effort has been spent in connecting tools to achieve the required traceability [13]. "The absence of standards has been shown to be a barrier to integration, as various tool developers remain unable to reach agreement on the appropriate point(s) in this space at which integration should occur. As a result, experience with tool integration has been largely at a tool-to-tool level, with little use of standard tool integration mechanisms." [1]

(2) Limited Extensibility

Commercial tools often do not provide open interfaces and well-documented APIs. Many research approaches in GSD are based on the automatic gathering of data to apply empirical metrics. This requires a variety of data to be collected from multiple data sources [5]. These data sources, e.g. a task repository often, have to be instrumented by the researcher. Other research approaches require the enhancement of the existing functionality of a tool, e.g. to provide decision support to the project participants.

The above mentioned two core issues and their implications often force the researcher to implement a research prototype from scratch, which is tailored to the scope of their research approach (e.g. [9], [10], [11], [14], [15], [16]). The requirements of these proprietary research implementations often overlap. For example a central and collaboratively accessible repository for the used artifacts of a tool is often required due to the distributed character of GSD. SCM capabilities of this repository allow recreating every state of a project for analyzing purposes.

The reimplementation of these features with the usual time and cost constraints often results in poor reliability and usability. As a consequence this often leads to a lack of motivation for the project participants to use it and results may be distorted. A high quality of the implementation is also an inevitable prerequisite for an industrial case study.

This paper presents the requirements for a uniform platform for research in Global Software Engineering called "unicase". Unicase addresses the core requirements for tools in GSD projects to lower the effort and risk for researchers to implement custom

tool solutions. The core idea of the project is adopted from the Eclipse ecosystem. The goal is to share the effort to implement basic requirements among interested parties. In the following we give an overview over related work, the requirements for such a platform. We describe the ecosystem idea and present an exploratory prototype. Finally we discuss future work.

2. Related work

In this section we describe related tools and approaches and show how they are different from our approach.
In general we found that any commercial, close-source solution we looked at does not provide the necessary extensibility for most research topics. This finding is not very surprising since they were not built for this purpose. They only provide very limited support for enhancing their model. Additionally if any they only provide a small set of extension points to add custom behavior. Furthermore cost for licensing or other legal implication is sometimes prohibitive for use in research. In particular we looked at the IBM Rational Software Architect, IBM RequisitePro, Telelogic Doors, Borland Caliber, Microsoft Team Foundation Server and IBM Jazz.
On the other hand the available open-source and/or academic tools have very similar deficiencies. We looked at a range of tools and we will go into detail for a typical representative of every category we identified. To give a short summary there are two categories of these tools, the special purpose research tools and the meta-CASE or integration tools.
The special purpose research tools are geared towards a specific research topic and have been designed to very specific requirements. They are not designed for extensibility.
SEURAT [10] is a research prototype for a new approach to rationale management. While the functionality for maintaining rational is sophisticated, there is no support for collaborative work at all, which is a prerequisite for a case study in this area of research since rational management is a collaborative activity.
Fujaba [14] is limited to a specific system model only. It provides only limited collaboration support because its Software Configuration Management is text based [7].

The purpose of the Sysiphus project is to provide a uniform platform for research in software engineering. The platform has grown for over 7 years now. It has been the foundation to several research topics and case-studies ([2], [4], [9], [6], [7]), recently in projects with over 50 participants. Stability and Usability became a serious issue, in a survey of 30 users 65% considered Sysiphus to be a useful tool in the current implementation compared to 100% who considered Sysiphus to be useful if it was more usable and stable. Sysiphus unfortunately does not meet the usability, reliability and scalability requirements for a bigger (industrial) case study.
The available model elements of ArchStudio are again limited. Architectural artifacts and product lines are supported only [15].
The second category of open source and/or academic tools are Meta-CASE or integration tools. In other words they either are CASE Tools to design custom CASE-tools or they try to integrate case tools from different domains.
Marama (formerly Pounamu) [12] is a Meta-CASE tool that allows specifying and generating CASE-tools. Although it supports any kind of model element and provides mechanisms for generating visual editors it does not provide support for collaborative work. The artifacts are persisted to the local file system and it is up to other methods and tools to enable collaboration.

Moflon [13] supports the integration of models from different tools by facilitating traceability links among the models from different tools. However for the integration of n tools it requires a non-polynomial number of transformers and a linear number of adapters to be implemented for full integration. This is prohibitive for any research that spans models from multiple tools.

3. Requirements

We performed exploratory interviews during the SE 2008 and ICSE 2008 conferences among researchers developing tools evaluating various approaches in SE. We also collected qualitative feedback from our industrial partners. This chapter summarizes our initial set of core requirements for a software engineering research platform. Our next step will be quantitative evaluation.

Model Repository

The core requirement is a central repository, which stores artifacts. As they are part of the project model, we will further call these artifacts “model elements” ([4]). The repository is able to store these model elements. The expressiveness of the model is at the level of MOF or an appropriate subset of it, such as ECORE (cite). Adopted from Sysiphus [2] we initially classify the model of unicase in 4 parts (see Figure 1): (1) The requirements model contains model elements like Scenarios or Use Cases [3] and describes the system under construction in the application domain. (2) The system model describes the design using

artifacts like UML ([8]) classes or flow charts. (3) The collaboration model contains management artifacts like tasks as well as rationale models and annotations. The structure of a organization in terms of groups and members is modeled in the (4) organization model. The model is easily extensible. Research approaches often require new model elements or attributes to be added. This is even possible during project run-time.

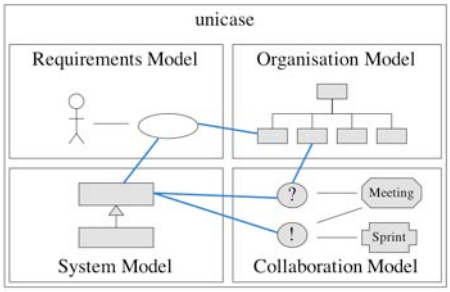

Figure 1: Linked elements from different parts of the model

Traceability

Related model elements in the repository can reference each other (see Figure 1). We call these references model links. Model links are typed. A task for example can be connected to a person with a link of the type "assigned to" expressing that the task is assigned to him. Model links are traceable. Unicase shall support the following two types of traceability [7]: (1) Intra-model traceability allows setting and following links inside a domain model, e.g. an association in a UML class diagram [8]. Inter-model traceability additionally supports links between domain models, e.g. from an issue [9] in the rationale model to a requirement which is the object the issue refers to. Furthermore, we distinguish three different kinds of traceability: (1) Vertical traceability is traveling through different levels of abstraction, e.g. from a requirement to a detailing use case. (2) Horizontal traceability usually follows inter-model links on the same abstraction level. (3) Temporal traceability allows tracing model evolution over time. This requires a model-based SCM described in the next section.

SCM and Distributed Collaboration

A precondition for tool research in GSD is support for distributed collaboration. One of the responsibilities of a SCM system according to [17] is workspace control. That is the ability to work on the same set of artifacts in different workspaces at different locations. Workspace control includes synchronization. Generally two different paradigms for control of concurrent modification of artifacts are well-known, pessimistic and optimistic concurrency control. Pessimistic concurrency control is often realized by locking and does not perform very well in large-scale distributed settings. An effective approach to workspace control therefore must build on optimistic concurrency control and as a consequence needs mechanisms for resolving conflicts that may arise.

The workspace interaction schema of unicase is checkout, update and commit as in popular source code repositories, such as SubVersion. This provides workspace isolation and users can work without being disturbed by other users' changes that may be preliminary anyway. Nevertheless the unicase platform additionally supports pushed updates to facilitate real-time collaboration if required.

An SCM system is also essential for many research approaches in terms of data collection.

Model Element Editor

Unicase provides three default views to browse and modify model elements. (1) The tree view allows users to hierarchically browse the whole model (see Figure 2). The hierarchy can be modified by drag and drop.

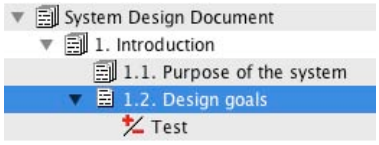

Figure 2: Example for a tree view

(2) The table view provides a tabular access to model elements and offers features like search and filter. (3) The model element editor view displays one model element and allows modifying it. Therefore it provides widgets for all standard attributes including text fields, text areas, checkboxes, date picker, but also widgets to set and modify references. The available widgets are easily extensible to add custom widgets like an impact probability matrix for risk management ([6]). As the underlying model is extensible, the editor is able to display any new model element "out of the box". Nevertheless the appearance and the arrangement of widgets is adaptable by the use of UI hints.

Graphical Visualization

Many research approaches require model elements to be visualized. Unicase provides a uniform draw pane, which supports visualizing and modifying model elements and links between them. Default visualization is available for any model element. Custom visualization can be added easily. The draw pane provides the features of a sate-of-the-art CASE tool diagram editor (see Figure 3). This includes resizing, arranging, coloring and zooming. The editor is

extensible e.g. for algorithms for complex diagram arranging.

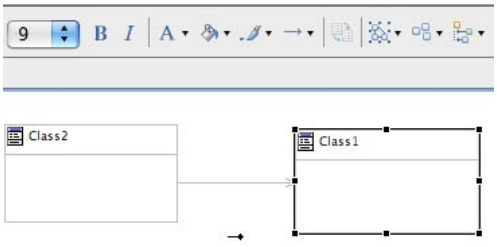

Figure 3: Example for a draw pane

Tool Instrumentation

As a research platform unicase provides comprehensive possibilities of tool instrumentation. The system is able to track when users read and modify elements, as well as which features they use. Custom events can be easily implemented.

Non-Functional Requirements

Unicase is easily extensible. Researchers are able to rapidly implement and integrate features for their research approach. The core system is reliable and is robust against the failure of extensions. Unicase provides a native look and feel as well as a common and natural usability concept. The common part of the framework is licensed under the Eclipse Public License.

4. Ecosystem

The idea of the unicase ecosystem is adopted from the Eclipse ecosystem. Instead of developing standard "infrastructural" software and libraries over and over again in different projects and at different companies, the idea is to collaboratively design, implement and maintain this software, thus significantly reducing the effort for every single participant. By "infrastructural", standard software we understand software that is not a competitive advantage in industrial terms or that is not part of the core research in academic terms.

We define the unicase platform as such software. The requirements mentioned before are common requirements of Software Engineering Tools we investigated. Many other tools have implemented similar features involving methods, techniques and algorithms that are well known and that have already been published. Implementation is feasible and does not require expert knowledge that is not widely available. So neither from an academic nor from an industrial point of view there is any advantage in restricting access to these.

From an industrial point of view such a platform provides cutting edge research on a stable platform. From an academic point of view a platform that is highly extensible and in use in the industry is a unique opportunity for industrial case studies with minimum threads to validity in terms of case study setup.

The ecosystem is open to anyone. Partners can only participate and use the developed technologies or they can actively contribute to the platform. By actively contributing the involved partners gain high impact on the requirements and the design of the platform.

The developed platform will be released under the Eclipse Public License that allows using any part of the platform in commercial or non-commercial products and even more importantly to develop proprietary possibly secret extensions upon it without the need to publish these extensions or their source code. This provides maximum flexibility for the platform use and re-use.

Currently we have two industrial contributors and one academic contributor apart from our chair. Our goal is to reach out for a larger community of contributors and participants - industrial and academic.

5. Exploratory Prototype

To demonstrate the feasibility of our approach we are currently developing an exploratory prototype. Based on this prototype we will implement an initial core of the system, which features the core requirements described in section 3 and then can be extended and modified by project participants. This section describes the design and implementation of the exploratory prototype.

We chose to build unicase based on the Eclipse Rich Client Platform (RCP). The platform provides several mechanisms to assure extensibility, stability and performance. Furthermore Eclipse provides a native "look and feel" as well as a mature concept for usability, which is widely established in the software development community. Most importantly however Eclipse features many existing frameworks and technologies especially in the areas of modeling and visualization. With its focus on software engineering tasks, it is a perfect match for several requirements of unicase, which otherwise had to be developed. Tools, which are implemented in Eclipse, can easily be integrated in the Eclipse IDE. We conducted a survey among users of the Sysiphus system, which meets comparable requirements to unicase (see section 2). 85% of the participants considered Sysiphus to be more useful if it was integrated in Eclipse IDE. Eclipse offers a mature plug-in concept. The plug-in mechanism makes it very easy to extend unicase by

adding new plug-ins. Furthermore this concept ensures reliability, as the so-called runtime platform is robust against plug-in failure. The exploratory prototype consists of a set of plug-ins described in the following, for additional information about Eclipse technologies see [18]:

EmfStore

The EmfStore plug-in is usually executed in a headless Eclipse environment and provides a repository for instances of the model defined in the model plug-in (see below). The EmfStore persists instances of the model with Teneo. Teneo is an object-relational database-mapping framework based on EMF and Hibernate. The EmfStore also keeps track of all model versions and allows clients to checkout a copy as well as to receive changes of the model. The EmfStore implements a change-based SCM as presented in [7]. Access control is based on a user, group and role model and the EmfStore is responsible for authenticating and authorizing user operations on the model upon checkout, update and commit operations. The EmfStore is capable of handling any model elements that are subclasses of the ModelElement class defined in the Model.

Model

The Model plug-in defines the model for representing artifacts. The model is based on EMF and expressed as an EMF Ecore diagram. From the Ecore diagram we generate the model plug-in code following the EMF model driven development approach. To implement new model elements one can just modify the Ecore diagram using a graphical editor and then regenerate the model. The regenerating the model will neither require changes to the UI nor to the server, thus making the implementation of new models a straightforward task.

The model also features a model instance generator that will generate a full-blown instance of a model with a configurable number of model elements, document width and height and a random seed. The generated model instances are a basis for extensive integration testing.

Model Element Editor

The goal of the generic editor is to meet the requirement “model element editor”, that means basic features for browsing and editing the model. Therefore it provides three views: a tree view (see Figure 2), a table view and a detailed model element editor view (see section 2).

To implement the tree view we used the Common Navigator Framework (CNF) (see Figure 4). This is the standard component used in Eclipse to browse trees like file systems or packages. To populate and visualize tree and table views we used the emf.edit framework, which is automatically generated for any custom model.

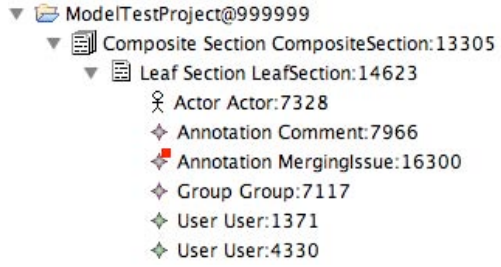

Figure 4: Generated project in the tree view

Existing solutions for model editors have at least one of the following four issues: (1) They are not usable to edit large models (e.g. the properties view), (2) they are not able to modify one element of the model but are build to edit the whole model at once, (3) They are bound to a specific model or require additional information about the desired visualization.

Therefore we implemented a new solution visualized by the Eclipse Forms technology. The editor uses the property descriptors generated in the emf.edit framework to reflectively build up the GUI at runtime. Basically it places a widget on the main form for every attribute of the EMF object using emf.databinding for synchronization between model and UI. Thereby it provides a view to edit a single model element without explicitly knowing its type and without any additional modification by the developer (see Figure 5).

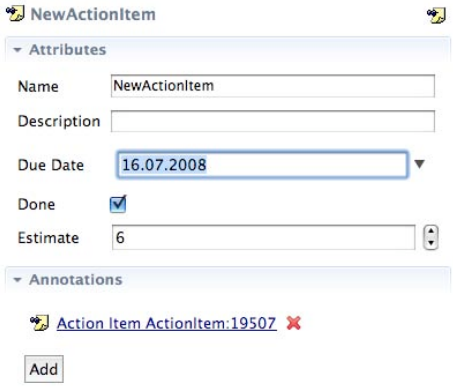

Figure 5: Reflectively generated editor view

Draw Pane

To provide a graphical visualization of the model we used the Graphical Modelling Framework (GMF) (see Figure 3). The purpose of GMF is to map EMF

models to visualizations of the Graphical Editing Framework. This allows to effectively generate custom draw panes. We had to modify the standard GMF editor to use an abstract model source instead of saving diagrams only to files. This was achieved by implementing a custom IResource.

6. Future work

We are currently implementing a base version of unicase. The first release is planed in October 2008. The framework will then be evaluated in student project as well as with our industrial partners. There are two initial research approaches we try to evaluate: (1) What is the impact of changes in the project model and how can changes be propagated to project members effectively? (2) How can the applied Software Lifecycle Model be adopted to mitigate process related risks?
Our next goal will be to allocate new partners for the project both in the academic and the industrial field. Further we plan to conduct a qualitative survey at ICGSE 2008. The target group for the unicase platform includes GSD researchers especially in the field of decision support who require tool support. The survey is meant to enhance and prioritize the requirements set for unicase.

10. References

[1] Anthony I. Wasserman, *Tool integration in software engineering environments,* Springer, Berlin, 21.1.2006.

[2] B. Bruegge, A. H. Dutoit, and T. Wolf. Sysiphus: *Enabling in formal collaboration in global software development*. In Proceedings of the First International Conference on Global Software Engineering, October 2006.

[3] B. Bruegge, Object-Oriented Software Engineering – Using UML, Patterns and Java, Pearson, Munich, 2nd edition, 2004.

[4] T. Wolf. *Rationale-based Unified Software Engineering Model.* Dissertation, Technische Universität München, July 2007.

[5] F. Shull and R.L. Feldmann, "Building Theories from Multiple Evidence Sources," *Guide to Advanced Empirical Software Engineering*, 2008, pp. 337-364.

[6] J. Helming, *Entwicklung eines Werkzeugs für das eingebettete Risikomanagement*. Diploma Thesis, Technische Universität München, Dezember 2006.

[7] M. Koegel. "*Towards Software Configuration Management for Unified Models*". ICSE CVSM'08 Workshop Proceedings, p. 19-24, May 2008, Leipzig.

[8] OMG Unified Modeling Language Specification Version 2.0, May 2004.

[9] T. Wolf, A. H. Dutoit, "A Rational-based Analysis Tool", 13th International Conference on Intelligent & Adaptive Systems and Software Engineering, Nice, France, July 1-3, 2004.

[10] J. E. Burge, David C. Brown, *SEURAT: Integrated Rationale Management,* In Proceedings of International Conference on Software Engineering, p. 835 ff, May 2008, Leipzig.

[11] A. Lucia, R. Oliveto, G. Tortora, *ADAMS Re-Trace: Traceability Link Recovery via Latent Semantic Indexing*, In Proceedings of International Conference on Software Engineering, p. 839 ff, May 2008, Leipzig.

[12] J. Grundy, J. Hosking, J. Huh and K. Li, *Marama: an Eclipse meta-toolset for generating multi-view environments*, Formal demonstration paper, 2008 IEEE/ACM International Conference on Software Engineering, Liepzig, Germany, May 2008, ACM Press.

[13] C. Amelunxen, A. Königs, T. Rötschke, A. Schürr: *MOFLON: A Standard-Compliant Metamodeling Framework with Graph Transformations*, in: A. Rensink, J. Warmer (eds.), Model Driven Architecture - Foundations and Applications: Second European Conference, Heidelberg: Springer Verlag, 2006; Lecture Notes in Computer Science (LNCS), Vol. 4066, Springer Verlag, 361--375.

[14] S. Burmester, H. Giese, W. Schäfer, *Model-Driven Architecture for Hard Real-Time Systems: From Platform Independent Models to Code,* Proceedings of the European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA'05), Nürnberg.

[15] E. M. Dashofy, A. van der Hoek, and R. N. Taylor, *An Infrastructure for the Rapid Development of XML-based Architecture Description Languages*, In Proceedings of the 24th International Conference on Software Engineering (ICSE2002), Orlando, Florida.

[16] T. Reinhard, S. Meier, R. Stoiber C. Cramer, M. Glinz, *Tool Support for the Navigation in Graphical Models*, In Proceedings of International Conference on Software Engineering, p. 823 ff, May 2008, Leipzig.

[17] J. Estublier, D. Leblang, A.van der Hoek,
R. Conradi, G. Clemm, W. Tichy, and D. Wiborg-Weber. *Impact of software engineering research on the practice of software configuration management.* ACM Trans. Software Engineering Methodologies, 14(4):383 430, 2005.

[18] Eclipse Projects,
http://www.eclipse.org/projects/listofprojects.php, June 2008

ICGSE Workshop 2008