Efficient Maintenance Support in Offshore Software Development: a Case Study on a Global E-Commerce Project

Zheng Yan
Helsinki University of Technology
Software Business and Engineering Institute
POB 9210, FIN-02015 HUT, FINLAND
Zheng.Yan@hut.fi

Abstract

Software maintenance is a very important phase in software development. It generally occupies the most of development life cycle in order to ensure software quality. This paper takes an e-commerce project as an example to study how to efficiently provide software maintenance support in offshore software development for a global deployed software product. Through interviews and a survey to the project developers, authors summarize the good methods and approaches used in its maintenance that greatly helped its success. Meanwhile, the authors also study lessons that influenced its efficient maintenance (e.g. extra workload caused by performance tuning, troubles due to sharp time-difference, problem-reproducing difficulty caused by testing environment difference and slow code transfer). Suggestions for further improvement are also proposed based on real experiences in order to benefit similar software development in the future.

1. Introduction

Offshore software development generally means that software is developed through collaboration of a team in an emerging country. It is one type of distributed software development that is adopted by many companies [2]. Lower cost, plentiful skilled staffs, high quality and trustworthy are main attractions for software development abroad. However, the offshore software development also faces difficulties and risks on decision-making, coordination, execution, communications and project management. There are many issues worth studying in the offshore software development regarding how to overcome its difficulties and reduce its risk. Amorbieta et al. [2] and Muller et al. [7] discussed how to make a decision on the offshore development, and how to choose right partner, and successfully start, organize, manage and execute this kind of projects.

Moreover, considering the software development, maintenance is a very importance phase, which generally occupies most of the software development time. It is a necessity in order to ensure sound software quality. When an offshore-developed software is used all over the world, it becomes more difficult for an offshore software development team to support the essential maintenance when code development is over.

Nowadays, seldom work studies software maintenance's influence on the offshore software development regarding the issues mentioned above. This raised a number of doubts from our literature study, comparing to our industrial experiences. For example, we believe that the challenges in offshore software maintenance may also affect the project decision-making and execution. How to provide efficient maintenance support in the offshore software development could be a big challenge worth special study.

Regarding the maintenance, some existing results need further study. For example, some work indicated that round-the-clock development is one of the advantages that benefit distributed software development by making use of the time zone difference [4-6]. It is worth further studying whether time difference can really benefit or retard the offshore software maintenance, because we experienced a lot of troubles to overcome the time difference in an offshore software development project that will be studied herein.

Culture liaison was introduced as a great help for alleviating distance and leveraging resources in [3, 7]. Are culture difference and understanding difficulties the only demand for a liaison role? What is the real need in terms of the efficient offshore maintenance? These questions are also worth studying, especially based on real cases.

Generally, the projects that require limited interaction with customers and have low strategic importance and high market capacity are treated suitable for the offshore development in [2, 7]. However, for a global e-commerce project that has more additional challenges than other software projects [1], but developed successfully offshore like the case we will study herein, good approaches used and challenges or lessons learned in its maintenance are

especially worth studying in order to extend the theory of offshore software development.

All of above are motivations of this paper. In this paper, we will take a global e-commerce project (GEC hereafter) as an example to analyze the reasons behind its great success and problems/lessons that are worth learning for future offshore software development. The focus will be on software maintenance: how to efficiently support the software maintenance in offshore e-commerce software development.

2. GEC overview

GEC was a web-based service for a global company's partners to order various company products. It was installed at a number of fulfillment sites to support product ordering from any country in the world. It was believed as the biggest B2B e-commerce system in 2000. This system greatly reduced ordering cost, tremendously improved the efficiency of ordering procedure and provided great convenience for both the company and its partners.

GEC provided global automatic management on products ordering, processing and order-maintaining for one of the biggest global companies in the world (customer company hereafter). Its main software was outsource-developed at a company in Singapore (provider company hereafter) during 1998-2001. The provider company completed the GEC software development and maintenance support on totally eight product versions, until the system was very stable and most features' implementation had been done. The system is currently maintained and enhanced by the customer company.

The GEC project was a project executed at different places all over the world. There were totally about fifty persons involved into this project at the provider's company including a development team and a testing team. Figure 1 shows its execution map. At the GEC software maintenance phase, the cooperation among different teams located at different places was needed in order to solve a problem.

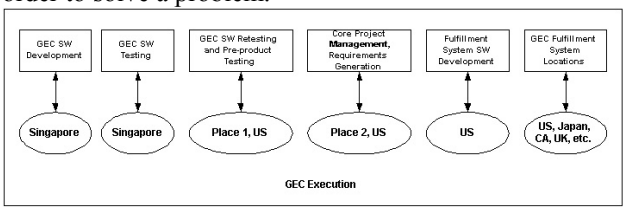

Figure 1: GEC project execution map

The GEC maintenance included several phases after the code development was finished. In this paper, we focus our discussion on the software maintenance conducted at Singapore. Figure 2 illustrates the maintenance phases of the GEC software.

The first phase of maintenance was conducted after the code was built and installed at the local test server in the provider company. The second phase of maintenance was conducted after the local testing passed. The build was uploaded to the testing server located at Place 1, US. The customer company's testing team there retested the software. The third phase of maintenance was started after the build was installed at the pre-product server at Place 1, US. The testing team of the customer company continued the test on more complicated use cases. The forth phase of maintenance was also required if there was any problem found in the product used by the GEC users all over the world. Generally, the product problems were emergent and required to be solved immediately because they directly affected the customer's business.

If there was a problem found in the maintenance phases, the testers either in Singapore or in US reported it using a team-connection tool. The development team could check and reply to the problem report after the fix was done. The team-connection tool managed all problem reports and processing history. Generally, the maintenance work was conducted in parallel with new version's development; especially the third phase and forth phase maintenance.

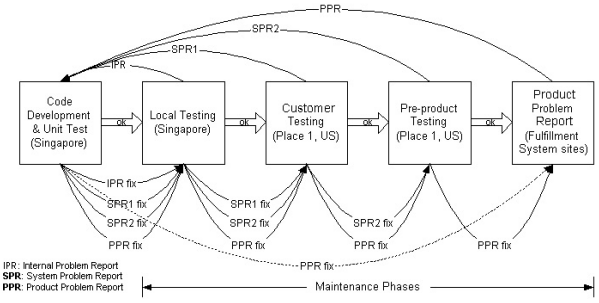

Figure 2: Maintenance phases of GEC software

3. Research questions and methodology

GEC was a successful B2B e-commerce system that brought a lot of benefits for the customer company's global sale. What we intend to study herein is the merits that benefited the GEC maintenance and the lessons that influenced the maintenance efficiency, as well as the aspects worth further improvement. This is because software maintenance support is one of the crucial aspects that influence the whole project's success. In addition, we also aim to clarify the questions mentioned in the introduction through the case study on GEC.

In order to conduct our research, we designed questionnaire and distributed it to all GEC developers for their feedback. The questionnaire was designed based on the first author's personal experience in the GEC development. The first author participated the GEC's development and maintenance on most versions as a

component leader. We received pieces of feedback. The questionnaire includes three parts:

- Participator's basic information regarding personal role inside the GEC project and contributions: Based on this part, we can identify the importance of response.
- Factors that affected the maintenance success: We asked the participators to mark the importance of those factors that we thought benefited the GEC maintenance.
- Potentiality for further improvement: In this part, we tried to propose questions in order to study over-time hard work's influence on maintenance efficiency, the reasons of extra maintenance work caused by performance issues, opinions on the difficulties of maintenance support in the GEC, and the reasons that caused the maintenance delay, as well as the hardness of code transference from old responsible person to the new one and from the provider company to the customer company.

Apart from the questionnaire, we also telephony interviewed several GEC developers. These interviewees are software component leaders from whom we can get to know all software components' maintenance information. One of them is the only person experienced all versions' development and maintenance. The main questions asked in the interview are shown in Table 1.

Table 1: Interview questions

1	Which event you experienced in the GEC maintenance gave you deepest impression?
2	What do you think the worst aspects that greatly influenced the efficient maintenance in GEC?
3	What do you think the good methods or approaches used that benefited the GEC maintenance?
4	What do you think the main reasons that delayed or benefited your maintenance work?
5	What is the reason that caused performance issue?
6	What is your opinion on improving the efficiency of GEC maintenance? What are your suggestions?
7	What do you like in GEC? What do you dislike?

In the telephony interview, we tried to get direct feedback on advantages and disadvantages experienced in the GEC maintenance. Especially, we got to know the interviewees' personal opinions on further improvement in order to overcome those bad factors that greatly influenced the maintenance efficiency. Each interview lasted for more than one hour. The interviewees provided valuable answers for each question. Through interview to them, we got a complete perspective on the whole GEC maintenance work.

4. Results

The results we got from the questionnaire and interviews are studied and analyzed as follows.

4.1 Factors benefiting successful maintenance

Based on the questionnaire and interviews, we specified the factors that benefited the GEC maintenance as follows.

It seems that the most important factor for efficient maintenance was attitude and relationship between the development team and its customer. With good attitude and relationship, mutual understanding was easier to build up in order to make trade-off on many issues at both sides. For example, the customer could be easier to understand the reasons of delay on problem solving if they knew the barriers and the hard work at the remote site. The development team would more like to accept extra requirements in urgent and offered solutions as soon as possible.

Compatible development/maintenance environment and efficient communications with the customer were also very important for efficient maintenance. But the provider company's maintenance environment was not perfect to support efficient maintenance. Both testers at US and Singapore did not share the same testing-system Database. This was one of the reasons that made it hard to reproduce the same problem in Singapore, but reported by the tester in US.

The tools such as project pager and approaches (e.g. sending liaison engineer and time-shift work) also played an important role for the efficient maintenance. The project pager was used in project emergency cases. For example, if there was a big problem found in product, the GEC help desk called the project pager. The pager taker at Singapore should call back and get to know the request for urgent maintenance support, even thought at midnight. On the other side, the provider company generally sent a liaison engineer to place 1, US for on-site maintenance support after the first maintenance phase. In addition, a development engineer was also arranged to work at Singapore nighttime, but daytime of US, to support immediate maintenance.

Since the project was big, it was impossible for one person to know all components of the whole system. Generally, it was hard for one liaison engineer and one time-shift developer to solve various problems raised at Singapore nighttime. Generally, they tried to look at the problem, but without any sense to solve the problem. They tried to console the customer until the next morning. They delayed time in order to let the responsible developers have enough time to rest at night, so that they could work efficiently next day. At the same time, they consoled the customer by giving them some feedback to make them feel that the problem was processing at the remote site.

Expert support was also very essential for performance issue. For example, the provider company lacked experts on DB2. The DB2 query caused a lot of performance problems since the database structure is very complicated in order to support the customer's business logic. In order

to help the development team, the customer sent a DB2 expert to Singapore. The face-to-face discussion effectively helped the performance tuning work at the maintenance phase.

Other factors that benefited the maintenance work are also important, but those are common factors for both distributed software maintenance and centralized software maintenance.

4.2 Improvement potentiality

Based on the results from the questionnaire and interviews, we summarize the lessons learned from the GEC maintenance in order to seek potentiality for further improvement.

4.2.1 Influence of over-time work. Over-time work was hated by all developers participated in the GEC, especially long-time over-time working (e.g. work from 10am in the morning to 3am next morning for two weeks or work over 3 hours every day for more than one month, which was normal during GEC maintenance). If working over-time, it is impossible to work efficiently and more mistakes may be made because of fatigue. But over-time working was generally forced to do, which happened mostly at maintenance phases if there were urgent problems to fix.

4.2.2 Reasons of extra maintenance work caused by performance issue. Performance issue found later on when the GEC had launched caused a lot of extra maintenance work. This kind of extra work sometimes greatly affected the whole project's schedule. The following reasons were pointed out as importance by the interviewees.

The first reason was that the software designers lacked experience on B2B e-commerce software. They had no much idea which aspects should be paid special attention in the design. GEC is one of the earliest E-commerce applications. It is also among the biggest ones in the world. At that time, no many people held real experience on such kind of software development. In addition, the platform APIs used for GEC development were not mature either.

The second reason was that the software designers at the provider company lacked concrete knowledge on real usage scale and system execution scenarios. Due to tight time schedule required by the customer, performance is not seriously considered at the software design phase.

The third reason was caused by the rapid growth of the GEC usage. The system scale was greatly enlarged within a short period. The initial success also encouraged the customer to deploy this system as broad as possible for its business partners all over the world. This raised many new requirements regarding performance improvement. The dynamic system growth was actually very hard to be anticipated at the design phase.

Herein, experiences were more crucial than technologies in order to avoid performance issue found later in the software product.

4.2.3 Difficulties for maintenance support. Based on the results of questionnaire and interviews, we found that the difficulties of maintenance support were generally caused by long-distance between the customer and the development team. The long-distance made face-to-face communication difficult, which further caused misunderstanding on the business requirements. It also caused time-difference, which, treated as beneficial for round-the-clock efficient software development, actually brought a lot of trouble in the GEC maintenance. The time difference made prompt support on product problem difficult and made it delayed to get feedback from the remote sites.

In addition, the product database was highly confidential. If the product database access was necessary for troubleshooting, the access duration issued was generally quite limited, which made the developers feel big pressure, not mention that the network connection was very slow. Limited accessible machines to the product system sometimes made trouble-shooters have to wait in a queue.

4.2.4 Reasons of maintenance delay. The main reason agreed by most developers about maintenance delay was the difficulties to simulate and re-produce the problem in the local environment. The database data applied for local maintenance were totally different from those for the customer's testing and were obvious distinct from the product. This caused a lot of trouble in reproducing the reported problems. Besides, the execution environment for development was different from the product execution environment. This was another reason made problem simulation difficult.

In addition, the difficulty to exchange idea regarding problems was also an important reason caused the maintenance delay. As commented by an interviewee, he sometimes had to wait until next day in order to get confirmation on some issue. If more discussion needed, longer delay might occur.

Apart from the above, developers' personal reasons and lack of project training might also cause maintenance delay. But those were not treated as so important.

4.2.5 Problems of slow code transference. Due to the frequent resource shifting in IT projects, it is generally impossible for one person to take charge of one software component in all life cycle of a software product. This introduces a practical symptom: responsibility transfer. The slow transfer also affected the GEC maintenance

seriously. This was mainly caused by the following reasons.

1. Lacking formal project training due to tight project schedule: The new responsible person is not so familiar with the project that he/she has to spend longer time to solve the problem.
2. Job competition: The old developer had pressure to be taken over by the new one. So he provided blur technical documents and code comments, and explained the code design carelessly. Similarly, the provider company also faced pressure if the customer withdrew the project.
3. No standard document format, coding format and design pattern deployed: This made new comers difficult to read and understand the code written by other people.

The above reasons also influenced the code transfer from the provider company to the customer's maintenance team after the contract was finished.

5. Discussion

5.1 Managerial implications

Based on the lessons learned from the GEC maintenance, we provide some suggestions for other offshore software projects.

Firstly, it is important design a series of working procedure in order to formalize the project management. It is necessary to make proper project schedule that saves some space for emergent events that may happen later on. Furthermore, it is also wise to make agreement with the customer regarding the solutions on emergency maintenance support, e.g. the accepted rules and policies for additional requirements raised from the product problems. In short, efforts should be made at the contracting phase to evade unnecessary argument that may occur in the maintenance phases. This is also a good approach to avoid hard over-time work that greatly affects the efficiency.

Secondly, it is crucial to pay special attention to performance issue and system scalability in the system software design. It is suggested to invite experienced experts to participate the design on related design issues and make instructions on software development regarding system performance that could guide the developers' programming in general. The customer should provide enough information to its partner about system scalability. It is suggested to provide a paper document to specify the maximum scale of the system, e.g. the size of some database table in the product, the quantity of a normal user's order request. With these approaches, extra performance tuning cost and work could be greatly reduced.

Thirdly, communication problem and time difference raised by the long distance is generally hard to overcome in the offshore software maintenance. It is better to introduce efficient communication tools for easy contact. Many literature studies have proposed a lot of good suggestions on this aspect [8]. But on-line communication or instant message is retarded by the time-difference. If efficiency on maintenance is more important, sending enough technical liaison engineers to the customer site is an effective method. But this may increase the travel cost. Those technical liaison engineers should be qualified enough to handle most of urgent system problems. One liaison engineer is impossible to know every aspect of a big project, so it is impossible for him/her to solve all kinds of problems.

Fourthly, It is better to provide as good as possible equipment to improve the remote access speed for remote problem solving and establish as similar as possible maintenance environment at the local development site. These will benefit problem re-producing.

Finally, It is essential to standardize offshore software project management and organization in terms of efficient code transfer. Project members should be trained for both project general purpose and their personal role purpose. A formalized project document template, coding template and design pattern should be introduced to the project members. This kind of training is a necessity in order to work out uniformed project software.

5.2 Comparison of own results to literature

Maintenance is a very importance phase in the software development. For the offshore software development, the maintenance brings a lot of challenges. Many challenges are actually caused by those advantages that people think could benefit the development according to the GEC experiences, e.g. round-the-clock development actually delays the maintenance; cost saving is normally not true at the maintenance phase because skilled developers are needed to work at the customer site in order to support on-site maintenance. Whilst the development site should also provide maintenance support as usual. The cost is obviously increased if hiring more people. If keeping the same resources, workload will definitely increase that will finally affect the efficiency of maintenance. All of challenges raised by the maintenance should and must be considered when the customer makes decision on outsourcing. The potential extra cost and difficulties that may be caused at the maintenance phase should be seriously considered and calculated at the decision making and contracting stages. Obviously, the maintenance related formal management should be involved into the offshore development management.

Based on our case study, we think it is more challengeable to provide sound maintenance support for globally deployed software product in the offshore development. The issues raised at the maintenance phase

are actually ignored in the current literature study. In Table 2, we summarize the research results based on the GEC experiences regarding the maintenance and compare them to the current literature.

Table 2: Research results and comparison to literature

Problems	Good solutions / suggestions	Literature study
Hard to build up mutual understanding to make trade-off on many issues at both sides	The provider keeps good attitude and relationship with the customer (This should be seriously considered at the decision making phase on partner selection.)	N.A. Trustworthy is not considered in [2, 7] for offshore software development
Hard to reproduce the same problem by the development team, but reported by the product users	Set up compatible development/maintenance environment with the product system, prefer as same as possible maintenance environment as the product system; provide sound equipments to access the product system for trouble shooting	N.A.
Maintenance delay caused by time difference	Set up efficient communications with the customer, e.g. making use of project pager for urgent maintenance support; sending enough technical liaison engineers to the customer site for local support; time-shift working in order to provide prompt support	Efficient communication tools are studied in [8]. However, no work proposed technical liaison engineers' great help and time-shift working for software maintenance
Troubles in maintenance raised by performance issue	Invite experts to the provider's site to cooperate with the development team for performance tuning issue; pay special attention to performance on the design; provide as detail as possible scalability description to the provider company; define programming regulations for better performance	N.A.
Hard over-time work that greatly affects efficiency	Design a series of working procedure in order to formalize the project management; consider urgent maintenance issues at contracting and project scheduling	N.A.
Long term code transfer internally and to the customer	Standardize offshore software project management and organization; train project members regarding formalized project document template, coding template and design pattern	N.A.

6. Conclusions and future work

In this paper, the authors studied the maintenance efficiency in offshore software development based on a real case study. According to the questionnaire and interview results, the authors summarized the good points that benefited the GEC maintenance and studied the bad sides that influenced its maintenance efficiency. In order to overcome and avoid those disadvantages experienced in the GEC, the authors further proposed several suggestions that could be referred by similar software development in the future.

Based on the practical experience and the GEC success, the authors believe big global e-commerce project can also be developed offshore although additional challenges need special consideration. The paper proposed some good solutions for potential problems that mostly have not been considered in the literature regarding the maintenance of offshore-developed software.

Since our work is only based on one real case study, the results achieved are only for reference purpose. Future work includes studying a set of efficient maintenance models that can be applied into various distributed software development. It is also significant to define a series of guidelines that could instruct maintenance agreement generation and execution.

References

[1] Alan R. Hevner, Rosann W. Collins, Monica J. Garfield, "Product and Project Challenges in Eletronic Commerce Software Development", *ACM SIGMIS Database*, Volume 33 Issue 4, December, 2002.

[2] Amorbieta, I., Bhaumik, K., Kanakamedala, K. & Parkhe, A., "Programmers Abroad: A Primer on Offshore Software Development", *The McKinsey Quarterly*, No.2, 2001.

[3] Battin, R.D., Crocker, R., Kreidler, J. & Subramanian, K., "Leveraging Resources in Global Software Development", *IEEE Software*, March/April, 2001.

[4] Dedene, G. & De Vreese, J.-P., "Realities of off-shore reengineering", *IEEE Software*, Volume: 12 Issue: 1, Page(s): 35 –45, Jan. 1995.

[5] Herbsleb J.D., Mockus A., Finholt T. A. & Grinter R. E., "An Empritical Study of Global Software Development: Distance and Speed", in *Proceedings of the 23rd International Conference on Software Engineering*, July, 2001.

[6] Mockus, A. & Herbsleb, J., "Challenges of Global Software Development", in *Proceedings of Seventh International Conference of Software Metrics Symposium METRICS 2001, IEEE*, pp182-184.

[7] Muller R., Ruland, D., Hoch, D., & Klosterkemper, B. "Offshore Software Development in Emerging Countries", McKinsey & Company, Articles volumn one – IT Management.

[8] Herbsleb J.D., Mockus A., "An Empritical Study of Speed and Communication in Global Distributed Software Development", *IEEE Transactions on Software Engineering*, Volume: 29 Issue: 6, June 2001, pp481-494.