Reorganizing an Offshore Software Project With the Goal of Favoring Knowledge Transfer

Carlo Consoli
IBM
Roma, Italy
carlo.consoli@it.ibm.com

Paolo Rocchi
Università LUISS
Roma, Italy
procchi@luiss.it

Paolo Spagnoletti
Università LUISS
Roma, Italy
pspagnoletti@luiss.it

Pietro Nico
IBM
Roma, Italy
pietro_nico@it.ibm.com

***Abstract*—This paper addresses the management of a software project developed by two groups of professionals, one working locally and the other one working off-shore. After the startup period lasting nearly a year, the project leaders observed that the quality and quantity of the software modules produced by the two teams were not up to expectations while costs had grown up. The project leaders analyzed the types and the amount of software tests required to ensure the quality of the software product. Finally, the management found out that the knowledge transfer process was the real root-causes of the project downfall. The leaders established a new organization as the solution to this problem. They replaced the two large teams with small groups to make communication and cooperation amongst people easier. Immediate evidence has demonstrated the effectiveness of this arrangement.**

Keywords- large software project; project management; offshoring.

I. INTRODUCTION

The basic aspect of offshoring is the notion that some jobs are movable [1]. It may be said that movable jobs are those with little face-to-face customer contact and with high information content. In relation to customer contact, Blinder and others use the term "personally" delivered or "personal" services to describe tasks that require customer contact or physical presence and "impersonal" services to describe tasks that have neither of these prerequisites [2]. In terms of high information content, considerable attention is paid to jobs based on Internet connections, which have greatly reduced the transportation costs of information [3].

The concept of offshoring started in the late 1980s when technology firms discovered emerging countries as basins of untapped resources of high-tech professionals at substantially lower labour costs [4]. Technology firms and Information Technology (IT) departments began to create Offshore Centers of Excellence (OCEs), which, in their early beginnings, related to assisting IT customers and later were devoted to more complex jobs, such as software development and maintenance.

OCEs, in many cases, evolved from a tactical to a strategic role [5]. As OCEs matured in the strategic role, they provided a higher degree of business value, the end goal being to operate in a seamlessly integrated model with the parent organization. However, the model of distributed software development sometime has become a critical success factor in the present global economy [6].

This paper focuses on a crucial aspect of offshore IT projects: the Knowledge Transfer Process (KTP). Issues related to KTPs frequently emerge within the context of IT outsourcing environments and several empirical researches have examined how the development knowledge needs to be shared among technicians and customers, and the quality of the exchanged information must be assured [7] [8] [9] [10]. Many focus on the customers viewpoint, instead [11] approaches the issues from the typical perspective of an offshore software supplier. Lee and others conduct a survey which illuminates the cultural differences which affect the performances of joined Western and Asian software development teams [12].

The present case study deals with this kind of cultural discrepancies and begins with an overview of the software project. Then, it illustrates how the issues rose and finally we shall explain the solution and its validation in relation to human communication and cooperation.

II. SOFTWARE PROJECT PLAN AND ORGANIZATION

An outsourcing project was undertaken by IBM (herein called the "IT Provider") to develop a ticketing application for an Italian company which transports goods and passengers (herein called "XYZ" or the "IT Client"). The aim of the project was to redefine the entire ticket trading system of XYZ, based on on-line sales and ticket offices deployed throughout the Italian territory.

The project began in late 2009 and its high degree of complexity required setting up a specific organizational model for the IT Provider which was deeply integrated with

 ISBN: 978-1-61208-367-4

the structures of the IT Client. This arrangement ensured that the customer and the software producer can cooperate in reaching common goals. Brief profiles of the IT provider entities are given below (see Figure 1.).

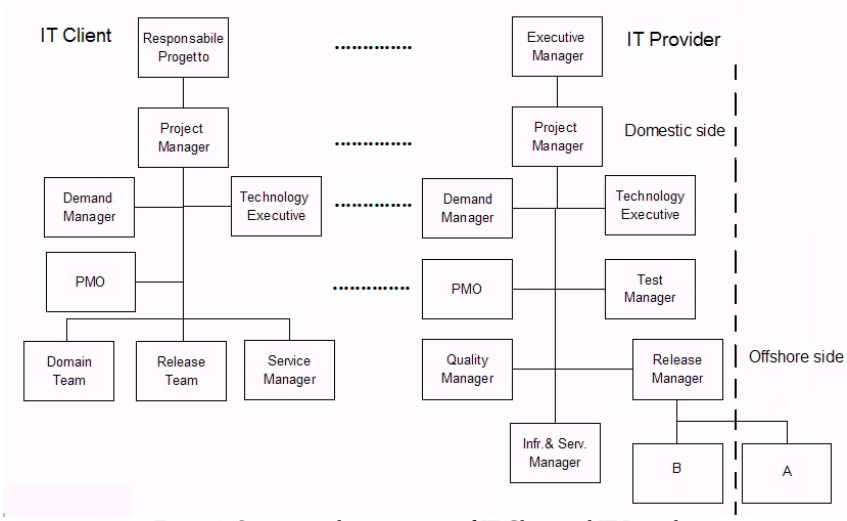

Figure 1. Symmetrical organization of IT Client and IT Provider

- The *Executive Manager* was formally responsible for all the aspects of the work and for the relations with the customer.
- The *Project Manager* was responsible for technical questions.
- The *Technology Executive* was responsible for the overall architectural design of the solution.
- The *Demand Manager* defined the detailed requirements and ensured the adherence of the solution to the customer's needs.
- The *Test Manager* was bound to the level of service.
- The *Project Management Office* (PMO) defined and maintained standards for project management within the organization.
- The *Quality Manager* was responsible for quality assurance.
- The *Release Manager* was responsible for the development and release of the software application, in charge of two different software development teams, one in Italy and one in India.
- The *Infrastructure & Service Manager* ensured that the infrastructure services met requirements in terms of size, performance and availability of the system.

We shall call one or more of the above managers "project leaders". Operations were carried out by the following groups of software practitioners:

(i) The *Release Team* brought out the various modules of the software application supplied to XYZ. This entity registered all the functions implemented and tested in the *Release Note*, a centralized platform used for software management. The Release Note was recognized as the official data handler and on request provided and still provides statistics on the work in progress.

(ii) The *Test Team* included from five to seven testers. The role of this team was to develop and execute test cases, find defects and set the defect status on the Release Note.

(iii) The *Development Team* was subdivided into the *offshore team* A and *domestic team* B.

The former included a variable number of programmers living in India: from 50 to 70, depending on the work load. Five team leaders from Italy managed team A and operated as the front end of the Italian development team B, which included 20 developers primarily involved in the analysis phase. The Indian team was chiefly in charge of coding and had a low level of responsibility.

Teams (i) and (iii) reported to the Release Manager; team (ii) reported to the Test Manager. During the year 2010 teams (i), (ii) and (iii) did not produce significant outcomes. It may be said that 2010 was a break-in period. In the early 2011 practitioners began to work steadily and the project leaders perceived significant difficulties. The quality and quantity of the software modules were not up to expectations and delivery times delayed. The project leaders established that the software modules should be tested in an accurate manner in order to check this unexpected downfall. In the mind of the managers, this control was even supposed to explain the low performances of teams A and B. Testing was executed in Italy, under the direct control of top leaders who surveyed the amount and kind of undertaken tests. They overlooked the number and typology of defects which are more telling on the technical plane, while the amount and kind of tests are appropriate for management purposes.

 ISBN: 978-1-61208-367-4

TABLE I. AMOUNT OF TESTS IN THE FIRST SEMESTER OF 2011

	B	**F**	**S**	**R**	**Total**
1 - Base Functions	73	63	0	0	4,599
2 - Billing Functions	16	6	5	63	411
3 - Advanced Tickets Purchase Options I	46	16	15	69	1,771
4 - Advanced Ticket Purchase Options II	72	29	24	85	4,128
5 – Ticket Purchase with Subscription	44	22	15	114	2,678
6 - Advanced Ticket Purchase Options III	48	90	16	136	6,496
				Grand Total	**20,083**

III. LARGE-SCALE TESTING

The *modules* are the basic components of the ticketing application; an executable version of the module is called a *build* (**B**). Specialists conducted two principal types of software test:

- They undertook **F** *functional tests* on each build. Functional testing focuses on recently implemented functions and overlooks previously implemented functions of the module.
- They executed **R** *regression tests* to validate the overall build, including new and old functions.

The Test Manager arranged **S** *sessions* per module to carry out the regression tests. More precisely, every module was submitted to a series of tests according to the following equations:

NFT (number of functional tests per module) = **B** × **F**
NRT (numbers of regression tests per module) = **S** × **R**
Total (total number of tests per module) =
= NFT + NRT = (B × F) + (S × R)

Table I exhibits data collected in the first semester of 2011. For instance, module # 6 – performing advanced functions in selling tickets – had 48 executable builds each of which underwent 90 functional tests. The manager arranged 16 sessions for module # 6 each of which included 136 regression tests. Thus, module # 6 had 4,320 (=48×90) functional tests and 2,176 (=16×136) regression tests; module # 6 had 6,496 (=4,320+2,176) tests in all.

The noteworthy values of NFT and NRT were basically determined by the amount of detected errors and strengthened the idea that something resulted in the low performances of the offshore and domestic teams. They settled to investigate this failure case by means of further inquiry

The states of the software defects were classified according to the rather usual triage as follows (Table II):

TABLE II. CLASSIFICATION OF SOFTWARE DEFECTS UNDER TESTIONG SESSIONS

New	Defect newly found by the Test Team.
In Progress	Defect subject to ongoing remedial work by the Development Team.
Pending	Defect pending remedial action while the Development Team gathers additional information.
Resolved	Defect remedied by the Development Team.
Reopened	Defect retested by the Test Team and found not to be remedied.
Closed	Defect tested by the Test Team and found to be remedied.

Defects classified as *In Progress*, *Pending* and *Reopened* will generically be termed *Open defects* hereafter. The Test Manager separately surveyed the new and open defects during the first semester of 2011. He noted that new defects were decreasing while the open defects were growing steadily from 50 (February 2011) to 130 (June 2011). This contrasting trend demonstrated that several software errors were causing cascade failures. The Test Manager meant to explore this negative phenomenon using the Release Note that is a software tool for monitoring the status of defects.

The Release Note provided a diagram that exhibits the six states listed in Table 2 (Figure 2); in addition the special block 'Release Note' indicates the status of defects just resolved and under registration by means of the tool Release Note. Teams (i), (ii) and (iii) responsible for handling precise states appear on the far left of Figure 2. For instance, the Development Team was in charge of the defects in the states: Pending, Resolved and In Progress. The flow diagram also shows the transitions of defects from one state to another with the transition frequencies. For instance, 2% of new defects evolve toward the Pending status.

The regular steps to handle a new defect are the following: New → Resolved → Release Note → Closed. But only 88% of new defects went to the status Resolved; 89% passed from Resolved to Release Note; and 84% of defects officially registered were closed. We obtain that a little more than half of the new defects were closed throughout the regular procedure

$$(0.88 \times 0.89 \times 0.84) \approx 0.65$$

In addition, note how 11% of resolved defects could not be registered by Release Notes (100% – 89% = 11%) due to various reasons. As many as 16% of resolved defects (see Release Note → Reopened), and 6% of closed defects (see Close → Reopened) were tested anew for partial corrections. A non-negligible amount of defects crossed the states In Progress → Pending; others followed the pathway: Pending → Resolved → In Progress. Essentially, the flow chart

 ISBN: 978-1-61208-367-4

provided details about the abnormal software production of defects and about the bad handling of those defects carried out by the Teams A and B.

The project leaders calculated the stability of the states in order to better understand the operations achieved by A and B. They counted the number of defects that remained in the same state from approximately February to May 2011 using a simulation program, and obtained the following results:

1) *New*: 4% of the new defects remained in the New state.
2) *In Progress*: 55% of the defects within this state remained so.
3) *Pending*: 92% of the defects within this state remained so.
4) *Resolved*: 11% of the defects within this state remained so.
5) *Reopened*: 1% of the defects within this state remained so.

It is worth explaining how these values – in particular 2) and 3) – do not derive from the priority of defects. Software defects with different urgency levels shared the same destiny. For example, a software error with high priority was revamped and closed in a short while; but a subsequent regression test often placed it into the Open status anew. This cyclic mechanism occurred more than once.

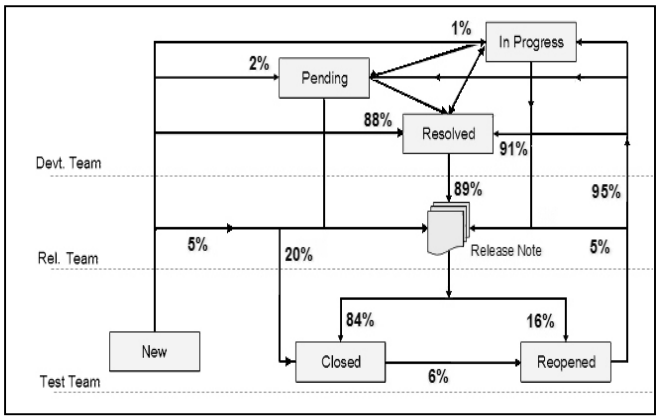

Figure 2. Flowchart of defects' states

Besides the numbers reported above, two practical observations on everyday job clarified the dimension of the project fail. Firstly, groups A and B were overloaded and spent most energy fixing the software defects rather than developing new code. Secondly, the high number of tests incurred time delays and high costs.

In conclusion, quantitative and qualitative data showed how the teams A and B turned out to be ineffective, in contrast with the high professionalism of individuals belonging to the two teams. The project leaders suspected that inefficient KTPs were heavily influencing the operations.

IV. ANALYSIS OF THE SITUATION

A special control group of experts searched for the reasons of this situation and discovered that all the root-causes were related to KTPs in a way. Communication and comprehension between members of the onshore and offshore teams were largely ineffective. In particular, the analytical report of the control group emphasized the following aspects:

I. The teams A and B had been arranged in two very different manners:

a) The Indian team was very large (50 to 70 programmers as described above) and rigidly structured according to hierarchical levels. There were *managers*, *general coordinators*, *area coordinators*, *specialized developers* and *generic developers*. They adopted standard methodologies; they used advanced software tools such as Rational but they followed somewhat rigid work-procedures.

b) Most of the team A members were young and lacking professional experience in large software projects. By contrast, the Italian team B included architects, analysts and developers with extensive experience and knowledge of the target market. The latter group took several details for granted, whereas the former group was completely unaware of technical requirements, the needs of the customer, the defects to correct, etc.

II. Testing was centralized in order to ensure full control of the software development. As a result, the Italian Test Team suffered an overload of activity which stressed the communication between domestic and offshore developers.

III. For team B, it was not a straightforward task to explain the requirements of XYZ and the Indian team. The latter had linguistic difficulties in reading some expressions typical of the Italian transport sector. There were considerable flaws in relation to the delivery of knowledge and knowledge acquisition by the Indians. The offshore team had very little domain knowledge and no understanding of how their development work fitted with the operations of XYZ.

IV. The coding activity of team A was managed by chiefs of the parent organization who viewed the offshore support merely as a low-cost production facility with an abundant supply of cost-effective labor for low-level activities.

Points I, II, III and IV taught the project leaders that difficulties could not be solved through limited counter measures. They decided to rearrange the structure of the entities involved in the project; in particular, they meant to improve the collaboration between teams A and B.

 ISBN: 978-1-61208-367-4

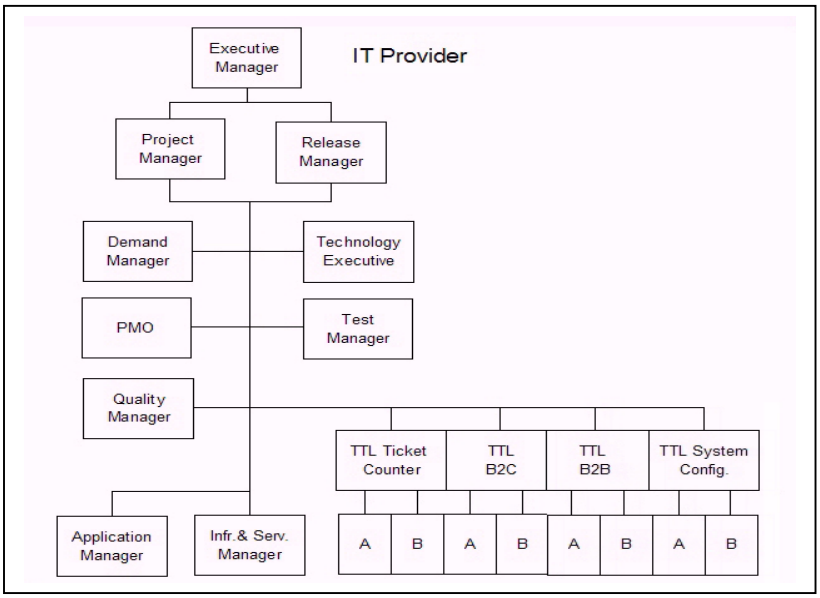

Figure 3. Renewed organization of IT Provider

V. WORKFORCE REORGANIZATION

The principal organizational changes are described as follow.

The domestic and offshore developers are subdivided into eight groups. That is to say, the ex-members of teams A and B comprise eight groups that are paired off and report to four Technical Team Chiefs (TTCs) (Figure 3). Each pair specializes in implementing a precise area of the ticketing application. The areas of railway ticketing are as follows:

- *Ticket Counter* = set of functions related to ticketing
- *Business to Consumer* (B2C) = set of Internet transactions that occur between the transport company XYZ and its customers
- *Business to Business* (B2B) = set of Internet transactions that occur between XYZ and other companies
- *System Configuration* = miscellany of technical functions

Italian and Indian developers become more tightly integrated as they have common and precise goals inside each group. In addition:

- Two Indian experts work in Italy to facilitate integration between multicultural and multilingual groups. This couple of people acquires knowledge of the needs and the characteristics of the Italian market through special training. They are wholly involved in optimizing communication between the domestic and offshore resources.

- The offshore team is assigned to carry out unit tests and functional tests in advance of the corresponding Italian team. A dozen Indian developers learn the ticketing methods of XYZ and are able to suggest corrections for the software modules in case of errors

- The entire testing process is monitored in a "*war room*" which includes experts from the onshore and the offshore side alike. The war room members monitor the status of a module, and analyze and evaluate issues in real time.

- The project leaders simplify the management of the defect states. Abnormal transitions are formally forbidden and, as a result, a negligible number of defects go into the Pending status.

Finally, the Release Manager relinquishes responsibility for software development and assists the Project Manager in gaining a better understanding of the progress of the overall software development.

VI. VALIDATION

In advance of the reorganization, a set of 18 principal functions were identified and scheduled by some project leaders who in addition calculated the resources required for testing these principal functions. Under the original scheme, each function should have required 406 tests of validation; this number includes all kinds of testing, from unit tests to regression tests. The test workload should have required 76

 ISBN: 978-1-61208-367-4

MDs (man-days) and should have caused 15 days of delay (Table III, upper row).

TABLE III. PROSPECTED AND DEFINITIVE AMOUNT OF TESTS REQUIRED BY 18 SOFTWARE FUNCTIONS

	Test per Function	*Total Test Number*	*Total MDs*	*Elapsed Time (days)*
Prospects	406	7308	76	15
Definitive Data	96	1728	18	4

Once the reorganization is completed, the 18 principal functions of the ticketing software applications are tested; the grand total of tests drops down from 7,308 to 1,728; the MDs comes down from 76 to 18 and the elapsed time falls from 15 days to four. This means that the new work organization carries out better software modules and in turn the number of tests necessary to ensure the quality of production decreases. As an example, the regression errors cease to exist. The human resources and the release times dropped down by up to one fourth of the resources previously supplied. A sound net 76% saving on costs and efforts was achieved.

VII. DISCUSSION AND CONCLUSION

This paper is intended to discuss a case of software development which was influenced by KTPs between domestic and offshore teams. The lessons learned by the project managers fit with some modern researches in the sense that KTPs can cause low performance, deprecable quality of software products, time delays and other noteworthy difficulties, while a unified and integrated solution that ensures perfect KTPs does not exist in literature.

It is worth noting that in the beginnings the teams A and B were classified as centers of excellence including skilled professionals. The situation was perfect on the surface and evident obstacles did not emerge in the first year of work. When problems cropped up, the project managers spent some time to discover the root-causes of the problems. Finally, the managers recognized that the cultural gap between the Italian and Indian developers and the diverging daily methods of work were the real origins of the economic losses and inefficient outcomes.

The present paper shows how the project managers have defined a novel governance structure to enable knowledge sharing across organizational boundaries of the off-shore environment. The new teams A and B are subdivided into four sub-teams and are guided by four specialized chiefs who ensure close communication amongst local and remote practitioners. People working in small groups can learn from each other about what is working better; they can get to know each other, keep discussion manageable and allow each discussion to happen in time. In substance, the introduction of small sub-teams turns out to be the organization key measure to enhance KTPs.

REFERENCES

[1] Prasanna B., Tambe Lorin M. Hitt - How Offshoring Affects IT Workers - *Comm. of the ACM* , 53(10), (2010), Pages 62-70.

[2] Bhagwati J.N., Blinder A.S., Friedman B.M. (eds) - Offshoring of American Jobs: What Response from U.S. Economic Policy? - *Proc. Alvin Hansen Symposium on Public Policy*, MIT Press (2009).

[3] Dossani R., Denny N. - The Internet's Role in Offshored Services: A Case Study of India - *Transactions on Internet Technology, Special Issue on the Internet and Outsourcing*, 7(3), (2007), Article No. 15.

[4] Lacity M.C., Willcocks L. - *Global Information Technology Outsourcing: In Search of Business Advantage* - John Wiley & Sons (2000).

[5] Lacity M.C., Khan S.A.,Willcocks L.P. - A Review of the IT Outsourcing Literature: Insights for Practice - *The Journal of Strategic Information Systems*, 18(3) (2009), Pages 130-146.

[6] Prikladnicki R., Audy J.L.N. - Process Models in the Practice of Distributed Software Development: A Systematic Review of the Literature - *Information and Software Technology*, 52(8), (2010), Pages 779-791.

[7] Mohamed A., Arshad N.H., Abdullah N.A.S. - Influencing Factors of Knowledge Transfer in IT Outsourcing - *Proc. 10th WSEAS Intl. Conf. on Mathematics and Computers in Business and Economics*, (2009), Pages 165-170.

[8] Faiz, M.F., Qadri, U., Ayyubi, S.R. - Offshore Software Development Models - *Proc. Intl Conf on Information and Emerging Technologies*, (2007), Pages 1-6.

[9] Prabhu N.V.A., Latha R., Sankaran K., Kannabiran G. - Impact of Knowledge Management on Offshore Software Development: An Exploratory Study - *Proc. Third Intl. Conf. on Advanced Computing*, (2011), Pages 121-128.

[10] Pilatti L., Audy J.L.N. - Global Software Development Offshore Insourcing Organizations Characteristics: Lessons Learned from a Case Study - *Proc. Intl. Conf. on Global Software Engineering*, (2006), Pages 249-250.

[11] Rajkumar T.M., Mani R.V.S. - Offshore Software Development: The View from Indian Suppliers - *Information Systems Management*, 18(2), (2001), Pages 63-73.

[12] Lee D., Smith A., Mortimer M. - Cultural differences affecting quality and productivity in Western/Asian offshore software development - *Proc. of the 3rd International Conference on Human Computer Interaction*, (2011), Pages 29-39.

 ISBN: 978-1-61208-367-4