Proceedings of the 10th WSEAS International Conference on COMPUTERS, Vouliagmeni, Athens, Greece, July 13-15, 2006 (pp826-830)

# Personnel Motivating Software Reengineering

DENISS KUMLANDER
Department of Informatics
Tallinn University of Technology
Raja St.15, 12617 Tallinn
ESTONIA

*Abstract:* In this paper we propose to use certain software projects' types as an additional personnel motivating factor. The personnel is a key factor in a company success, therefore management has to motivated workers, give them sometimes challenging and interesting work. Reengineering projects are providing a good chance for such interesting development since usually contain technological and algorithmic shifts and allow developers to expand their skills. The paper revises software reengineering projects, defines how those can be used to motivate personnel and what key benefits of that process are. Finally two companies' cases are described where the personnel motivating reengineering increased productivity of the development team and saved developers for companies since they decided to stay.

*Key-Words:* Personnel, reengineering, software engineering, software development

## 1 Introduction

Highly professional and motivated personnel are a key factor of the company success nowadays [1]. Unmotivated personnel can be seen as a major risk factor [4, 7] since workers either decrease their productivity or going to leave the company. In the last case the company could loose knowledge and time, which will be required to train a new worker to the level of the retired one. The software development is a highly technological sector [12] with a shortness of personnel resources in many countries at the moment and is requiring a lot of time to adopt new workers. Nowadays increasing competition between software vendors and much more demanding markets force companies to stabilize their productivity and improve their development process in all respects using all available techniques [9]. Personnel motivating projects could be a good method to achieve that into addition to other common motivating factors proposed so far [4, 5, 7].

Software reengineering projects are the best software development projects type to apply personnel motivating principles. Those usually contain technical shifts, unlike standard software development where programming language and techniques to be used are usually the same as in many previous projects. In the reengineering projects developers can learn something new, expand their skills and became enthusiastic about their work.

The paper is organised as follows. The section 2 briefly describes the software development work cycle. The following section concentrates on the reengineering projects revising their goals and strategies. The personnel motivating software development is described in the section 4. Here benefits of the proposed method are reviewed with some ideas on how to organise such development. Two companies cases where the proposed idea where applied are reviewed in the 5th section including achieved results. The last section concludes the paper.

## 2 Software Development Work Cycle

The software development work cycle has to be reviewed before the personnel factor can be considered. It is done to define unequivocally projects that are to be covered in this article. Besides the development process affects ways the discussed factor can be considered and treated.

There are a lot of models for software development and some of them are quite basic ones like the waterfall or spiral software development [3] methodologies. It is a bit hard to build a model that could adequately reflect all those on a general level, but the following very simple one should demonstrate basic principles of most of them. The model contains three parts: requirements gathering, then design, thereafter software (program code) and this can result in developing new requirements, so the cycle exists and a new development process is started. Notice that a new software development starts from the requirements formulation, which is the first step of the development process.

The software reengineering process is very similar to the previously described one and also starts from the requirements formulation. The main difference is that requirements are formulated basing on an existing software package to be rebuilt. Therefore the existing software (program code) is a starting point of the development cycle. In this software development case the source system is studied at the first step to find out existing processes, bottlenecks, the number and kind of documents in use and so forth. New requirements are

also gathered as it was done in the new software development process described above and then mixed up with the existing logic to produce new system requirements, then design and thereafter a new system.

# 3 Reengineering Projects Strategies and Approaches

This section is designed to review different strategies and approaches that are specific for the reengineering projects.

## 3.1 Reengineering goals

The software reengineering always starts from an existing software and makes a shift of that software to anticipate new requirements [6]. The following reengineering goals are most typical:

- Moving to a new platform. It is a technological shift that could include moving the logic to another platform in parallel to the existing version, rewriting the product using a modern programming language (the next version of the language used so far) and so forth;
- Rewriting to implement another business logic, add more features etc. This process is similar to implementing a new version of a product. Notice that in addition to the business logic change, some programming techniques can also be changed in scope of that project. Examples of that could be developing new sophisticated algorithms, programming using OOP principles, services and agents oriented programming etc.
- Combined: shifting at the same time more than one element: the business logic, the technical base and the algorithmic structure.

## 3.2 Personnel factor and different development strategies

There are two major strategies in the reengineering projects, which are general for all software engineering approaches. The first strategy asks to minimize time and resources spent on the development process. A project management in that case concentrates on the efficiency of a concrete package development. This strategy is usually applied in the following cases:

- A project cost should be minimum / less than usually to get a customer or certain sector of the market;
- There is no major shifts to be done in the development and the best development case is to be applied in this and future projects;
- A project is a standalone one and no future projects should be considered.

Another strategy concentrates on the overall development process optimization by considering future projects also. This strategy allows spending certain recourses on necessary experiments and adopting new technologies. The personnel factor is usually seen in that case as a restricting factor that should be overcame to ensure the project's success. In this paper we propose to see this factor from another point of view: the project should be a motivating factor for the personnel and this should be considered while planning and implementing the reengineering project. This detail, which looks to be very small, could be crucial for the project's success. Contemporary software engineering methods ensure general success of projects, but high competition asks to maximize this success by applying additional restrictions for used resources, spent time and so forth [9]. The general success is not enough any more to win the race, outstrip competitors. Therefore middle-size factors should also be considered to increase productivity and profitability of projects.

Another consideration is more internal for companies although is also indirectly caused by the business environment. There are a lot of countries where the shortness of skilled programmers affects a lot the development process. The personnel is an important part of a successful company and should be carefully used, grown and motivated to ensure the whole company success [10]. All technical equipment is easy to buy and it will work for you 24 hours, will not retire since would have other experience or because the work is started to be too boring.

# 4 Personnel Motivating Software Development: Benefits and Organization

There are a lot of developers that are working for a company during 4 or 5 years and then are changing the company since one of the following:

- They would like to obtain a new skill to ensure their future in the quickly changing technological world;
- They are tired to do the same things using the same tools;
- They would like to try other technologies and development principles.

Notice that major standard factors like friendly environment, salary and so forth are well-studied and are not considered in this paper because companies are already skilled in addressing those factors. At the same time developers are still changing workplaces, going sometimes to work-places with slightly smaller salaries

because of the above listed factors. The new technologies question and motivating personnel to stay in the company in scope of (or using) the reengineering process is considered by us as an important method to ensure companies success and stability. Skilled developers leaving a company create certain troubles for the company in major cases. Often it takes several months or years to train a person replacing the left developer and it decreases the company productivity. The more stable is workers group the more stable is the company. Therefore we see a challenge to motivate personnel when software is reengineered as an important tool to ensure company's profitability and ability to outstrip competitors.

In the following part of this section we will review benefits of the motivating software reengineering, how it can be incorporated into the development process, how it will affect it and what aspects should be carefully considered.

First of all let's return to the development work-cycle and review methods that are using an automatic code generation and evaluate the following question: "Is the discussed problem actual or it is artificial since developers will be completely excluded from the software engineering process soon?" Unfortunately a lot of software companies have troubles applying the automated code generation routines, which are rather specific to some unique projects. The first problem is that the development of a design on the low level, which is a requirement for the automatic code generation, requires a lot of efforts. It is the same complex from the amount of work point of view as the actual programming. Notice also that it is easier to find developers than to find the same number of designers. Another reason why developers are still presented is that they are able to identify a lot of problems in the design and they do that. The automatic generation programs are not intellectual enough so far to detect a missing code and do the "what–if" analyses.

That are some arguments having developers still in the project team and therefore collaboration between software designers and developers should also be carefully considered. The first important question is the design grain. It is quite hard to find a level of the design details that will illustrate completely the designed logic and will not be too complex to produce. There are much more developers than designers and they could and should support the design process by implementing their code and producing an efficient feedback to the designers about design errors. Besides involving into the design a set of new persons could help to find new ideas for the design.

Now, when it is clear that there are a lot of case when developers still remain in the company, advantages of personal the motivating development to be reviewed on the more detail level. Major benefits are the following:

- Developers will have new experience, increase their number of skills. This provides workers with confidence in their future, since in case they will have to relocate, company will fire them etc they will have up to date knowledge allowing finding a new position;
- Developers see their work as more challenging and interesting. It results in a higher productivity of developers and leads to establishing much more creative team. Involving developers into the technical design could gain additional credits to that team and will support developers wish to be efficient. This topic will be discussed below under the supporting development approach;
- In the result developers remain in the company eliminating the need to learn new workers and keep companies productivity constantly on a high level, which is a clear benefit for employers;
- Besides employers see how workers can adopt new technologies; are they willing and able to learn something new. Notice that inability to learn is not something bad. It just provides employer an information about potentiality of their teams in future projects and allows segment the team by their ability to learn. The following partition of personnel can be used:
  - Innovating;
  - Slowly changing / slowly learning;
  - Static.

  The segmentation process will allow keeping different people happy by addressing their requests and expectations in the best way. Static workers can be kept on fixed project that are not migrating to other technologies, while innovating should be used to start the technology migration process. Moreover this segmentation helps to plan the learning process as slowly learning developers need more time and this factor should be considered to avoid producing a stress situation for them;
- Software development motivating personnel in some cases can increase cooperation between team members, first of all between developers, as they need to discuss new techniques, may be learn something together in groups etc [9, 11]. Workers collaboration is a standard problem in many companies sometimes addressed by having external parties to establish contacts. The proposed method is one more instrument to make the collaboration between team members stronger;
- The proposed method addresses also workers fear to loose their workplace. In the described situation they will be able to see that management is willing to learn personnel instead of hiring new workers with

other skills. It leads again to better "ecology" in the company affecting productivity etc.

A process of establishing personnel motivating development requires certain activities to make it smooth and avoid negative effects that could arise. The iterational software reengineering versus development was discussed in many papers [2, 6, 8]. In the scope of this paper some modification of the iterational process to be proposed to address the motivation factor.

Iterational development means that a project is divided into parts (iterations) that address different features to be implemented. In our case we propose to start the reengineering process from iterations that include fewer features than usually and provide more time for the learning process. The number of included features should grow from iteration to iteration and reach the maximum in the last iterations, while time for the learning process will be decreasing. Notice that in general iteration could become to be the same size from the timescale point of view, which enables applying the agility methodology without additional expenses.

The learning process incorporated into the development process is an important factor of the project success. It is always positive to have any training including a training prior the project start, but training without experience is not very efficient. Therefore a better way from our point of view will be to have training during the development accordingly to features to be implemented in one or another iteration. It means that the learning process and implementation process should be synchronized and carefully planned, so developers will have learned or will learn all they could need to use during programming the current iteration and the more they learn with having possibility to try at the same moment, the more efficient learning it will be. Basing on our experience we have to note that such learning usually will not affect the development process from the spent resource point of view since developers apply their skills first time anyway slowly and inefficiently. Therefore learning and applying (including experiments) could be even better than learning and developing after one month. It is important to plan the learning process also accordingly to the segmentation of workers by ability to study. Individual plans for developers or groups of developers could greatly support the overall implementation process.

It is also crucial to obtain a support from the management. They are deciding to hire new personnel or learn existing workers. They have to understand all benefits of the motivating development process, provide enough time for the learning and adaptation process winning a time from incorporating into the company new workers.

The last but not least element of the supporting infrastructure is applying supporting software development principles. The classical software development work-cycle shows that any next step of the software development is done only if the previous one is completed. The supporting software development principle is defined as using each step to help doing correctly the previous one. It is achieved by establishing an efficient feedback cycle between neighbour development steps. This technique provides a better control over the project‘s progress, which is important when the new techniques are used. It also involves developers into the technical design stage allowing them either affect the design or better understand it, which increases their motivation as they see ways to apply their knowledge, improve products and consider this type of work as more challenging. A collaborative team that is always established in the scope of this methodology improves the knowledge transfer, communication between different team groups and increases productivity by eliminating certain time-lags.

# 5 Cases Study

Here some companies where the proposed approach was applied will be reviewed. Notice that the described approach is not targeted to fit all projects. At the same time a lot of companies could benefit from applying this methodology to stabilise the development process and improve their work environment. Unfortunately we are not able to provide full names of companies to be described due confidentiality reasons. At the same time those examples are quite typical and therefore those can be extended to many other companies, so the cases companies' names are not important.

## 5.1 IT department of an insurance company

The first case to be reviewed here is a reengineering project in an insurance company that is done internally by the IT department. The number of workers in the department was around 20. The insurance company deals with a variety of insurance products, but the main focus of the project was made on none-live insurance products, i.e. property insurance. The previous version of the product was released using very old programming language and required both technological and business logic shifts. The reengineering project was mainly started by a new management team. The main goal was to re-develop the company's IT system and not to destroy the IT team since they had very important knowledge on the existing system that had to be used during another several years. A pilot project was started to identify workers ability to learn and use new technologies. Developers were partitioned into several groups: static were mainly supporting the old system,

while innovating developers were included into the new system team. Later interviews have shown that 20% of developers had plans to leave the company but decided to stay since the new project was interesting and challenging to them.

### 5.2 Software company

The second case to be reviewed is a general software package produced by a software company. This product is sold to a variety of customers and its reengineering project was established to enable using it on another platform. The product was mainly organized to anticipate both customers and developers' needs to study something new. 10% of company developers have left the company looking for something more interesting and challenging during several months before the project was decided to go. It was a period when during last 7 years the company wanted to maximize the income by using a programming language that allowed developers producing packages very quickly. Another platform migration project was in plans during several years, but was not considered as an important one. The project was re-prioritized to provide developers possibility to expand their knowledge, learn something new and may be find new development ways for the main product. It was hard to convince the general management to start the product, but in the result a collaborative developers' work-team was created. It enabled the knowledge transfer between different teams and made developers much friendlier and open. This eliminated certain gaps in communication and increased productivity. Moreover after several years this platform became to be the main money-producer for that company.

## 6 Conclusion

In this paper software reengineering projects are proposed to be used as a personnel motivating factor. Here we mostly concentrated on developers, which are quite an important resource of software companies and IT departments. Technological and algorithmic shifts that can be implemented in the scope of reengineering projects can provide developers with new skills and make their work much more interesting and challenging. Personnel motivating projects requires proper planning by including a planned learning process into iterations, a support from the company management and establishing a supporting infrastructure like applying supporting software development principles enabling an efficient feedback. Workers can be partitioned into segments by their ability to learn. Each group should be adequately associated to projects and addressed during the learning process by plans that correspond to their abilities. All this results in an increased productivity and an improved communication and work environment and could lead to a worker's decision not to leave a company, which saves company knowledge and resources.

*References:*


[1] M. Armstrong, *A Handbook of Personnel Management Practice*, Kogan Page, London, UK, 1991

[2] A. Bianchi, D. Caivano, G. Vissagio, Method and process for iterative reengineering of data in a legacy system, *Proc. Working Conference of Reverse Engineering*, 2000, pp. 86-96

[3] B.W. Boehm, A spiral model of software development and enhancement, *Computer*, Vol. 21, No. 5, 1988, pp. 61-72

[4] D. Daly, B.H. Kleiner, How to motivate problem employees, *Work Study*, Vol. 44, No. 2, 1995, pp. 5-7

[5] B. Gerhart, How important are dispositional factors as determinants of job satisfaction? Implications for job design and other personnel programs, *Journal of Applied Psychology*, Vol. 72, No. 3, 1987, pp. 366-373

[6] J. Henrad, J.M. Hick, P. Thiran, J.L. Hainaut, Strategies for Data Reengineering. *Proc. Working Conference on Reverse Engineering*, 2002, pp. 211-220

[7] F. Herzberg, One more time: How do you motivate employees?, *Harvard Bus. Rev.*, Vol. 65, No. 5, 1987, pp. 109-120

[8] S.A. Kareem, V.V.S. Raveendra, Data migration using iterative methodology, *Proc. of the IASTED International Conference on Software Engineering*, 2006, pp. 206-211

[9] D. Kumlander, Software design by uncertain requirements, *Proceedings of the IASTED International Conference on Software Engineering*, 2006, pp. 224-2296

[10] B.P. Lientz, L. Larssen, *Manage IT as a Business: How to Achieve Alignment and Add Value to the Company*, Elsevier, 2004

[11] R. Ludlow, F. Panton, *The essence of effective communication*. Prentice Hall, 1995

[12] M. Rauterberg, O. Strohm, Work organisation and software development, *Annual Review of Automatic Programming*, Vol. 16, 1992, pp 121-128