Proc. of INSPIRE 98 International Software Process Improvement Conference in Education and Research, London, 87-98, 1998

Simulating Software Projects – An Approach for Teaching Project Management

P. Mandl-Striegnitz[1], A. Drappa[1], H. Lichter[2]
1 University of Stuttgart, Stuttgart, Germany
2 Aachen University of Technology, Aachen, Germany

Abstract

In this paper, the results of two industrial case studies on software project management are presented. These case studies revealed key problem areas in software project management and also proved that there is a serious lack in training of software project managers. To reduce these deficiencies, we suggest a new approach for teaching project management: simulating software projects. The simulation is based on models of software development processes that can be executed. We present a new simulation model, the QA model. The development of that model was strongly guided by the results of the case study. We show in detail how simulation models can be bolstered by specific industrial data to provide the highest possible benefit to simulation model users.
Keywords: Software Engineering, Project Management, Software Engineering Education, Empirical Investigation, Process Modeling.

1 Introduction

Over the past years, software development projects have increasingly been plagued by schedule and cost overruns while product quality is often poor. In reaction to this, many software companies have started initiatives to improve their software development processes [1]. In this context, a key factor in software development projects deserves particular attention: the project management.

In this paper, we present the results of two industrial case studies on software project management. These studies were motivated by the company's goal to improve software project management. At the beginning, a project management assessment was performed using structured, on-site interviews with software project managers. To verify the qualitative data collected during the assessment, a second investigation was performed by measuring quantitative data of a completed software development project. Analysis of the collected data clearly demonstrates a lack of education and training of project managers.

Consequently, project managers need to be better trained before leading software projects. Above all, management experience is required to plan and control software

projects successfully. Experience is the key to assess situations and to foresee consequences of possible decisions with some confidence. However, becoming an experienced software project manager usually "costs" many wrong decisions taken in real software projects. Within our research project SESAM (**S**oftware **E**ngineering **S**imulation by **A**nimated **M**odels), we propose a new approach for software project management education. The new solution consists in simulating software projects interactively. The basic idea is to create a model of the software development process and to execute the model using a simulation system. The model is complemented by an initial project scenario. Thus, software projects with a given task, given resources, or constraints can be simulated.

This paper introduces a new simulation model, the QA (**Q**uality **A**ssurance) model. The results of the two case studies thereby significantly supported model development. The QA model covers all software development activities from requirements specification to product delivery. It especially emphasizes the effects of quality assurance activities.

We will then show how simulation of SESAM models, e.g. the QA model, can help to reduce the deficiencies of software project management identified in our investigations. Vice versa, if planned according to the metrics system used for a SESAM model, empirical investigations can contribute to the validation of the model and help tailoring it to specific environments.

2 Two Case Studies on Software Project Management

2.1 Description of the Case Studies

At the beginning we performed a project management assessment by means of structured, on-site interviews with seven software project managers. Its purpose was to assess and better understand the current practice and problems of software project management. Furthermore their impact on software development projects was investigated. The questionnaire used for these interviews covers more than 70 factors potentially influencing the process and outcomes of software projects [2]. To develop the questionnaire, other investigations were taken into consideration [3], [4], [5]. The questions mainly address human factors and organizational aspects.

Most of the information collected during these interviews was qualitative and subjective. Hence, in a follow-up study we performed a quantitative evaluation by measuring an already completed software project in considerable detail to verify the results of the first study. Furthermore this increased insight into the reasons for certain problems occurring during software development projects. The design of the second study is mainly based on [2] and [6].

Measuring a software project requires multiple sources of information. Hence, the empirical investigation was implemented as follows: First we collected data on the basis of several interviews with the technical manager and the project manager. In addition, we asked the project staff to complete questionnaires concerning personal aspects or quality assurance activities. Some information concerning defect and change management data was provided by a tracking system used in the software

project. Additionally, we analysed the project key deliverables such as the project plan, project status reports, source code, and user documentation.

We now briefly describe the central aspects of software development projects we were interested in. Some of these aspects were only recorded for the project examined in the second study. The complete definition and planning of both studies can be found in [2] and [7].

2.1.1 Project Profile

To record the specifics of the investigated projects, we measured characteristic attributes such as project nature, project scope, and project type. Furthermore we examined items such as the organization structures used on the project and the application domain.

2.1.2 Project Totals

To get an overview of the performance of the investigated projects, we compared the planned and actual items for overall project effort, schedule, and cost.

2.1.3 Project Tasks

In both studies we asked the project managers to list all tasks that have been performed during the project and to estimate the percentages of project effort which each task required. To validate this information, in case of the second study we collected detailed task-by-task information on the resources, effort, schedule, and cost. Furthermore we were interested in the documents that had been produced in each task as well as in the size of the deliverables in terms of page counts.

2.1.4 Project Environment

To analyse the influence of the project environment, we were interested in attributes such as the goals or constraints levied against the project by senior management, customer or user demands. For both studies, the level of customer and user involvement during different phases was measured on a 5-point ordinal scale. In the second investigation, this information was validated by additionally measuring quantitative factors such as effort and schedule of customer and user involvement by tasks and for the total project.

2.1.5 Project Management

To assess the strengths and weaknesses of project management, we measured the following variables:

- Project management experience
- Time and effort invested by the project manager for management activities
- Methods and tools used for project planning, progress tracking, and cost tracking
- Responsibilities for project management activities
- Risk management (methods used for the analysis and monitoring of risk factors)

In case of the second study, this information was verified and complemented by analyzing the corresponding documents (e.g. project plan, status reports).

2.1.6 Project Staff

Since project staff skills mainly influence the software development project, we collected information on the qualification and experience of team members. Furthermore, to assess the quality of personnel management, we recorded data such as staff availability, training of team members, effort spent for communication, and team continuity. For the measured project, we were also interested in the size and structuring of the project team.

2.1.7 Quality Assurance

For the aspect of quality assurance we collected qualitative data on the kind of quality assurance function and the quality assurance activities performed during the project. For the second study, we explored the effectiveness of quality assurance activities. We collected information on the number of defects found and the effort required. For every defect we recorded the origin, the severity, the amount of effort required to fix it, and the quality assurance activity that helped to uncover the defect.

2.1.8 Project Outcomes

The project outcomes were only analysed in case of the second study. For each document produced during the project we were interested in the nature of the document (e.g. specification, design document, user documentation) and the number of pages in total. For the source code we measured the size in lines of code per baseline. We distinguished different code classes such as delivered code, generated code, prototyping code, test code, and reused code. In addition, the complexity of the source code per baseline was measured.

To capture the productivity, we were also interested in the number of documentation pages and in the number of lines of code produced per calendar month, and the number of personnel involved.

2.2 Results of the Case Studies

In this section, we present some of the findings resulting from our investigations. We concentrate on the most important weaknesses concerning project management. An overview of all results is given in [2] and [7].

2.2.1 Insufficient Time Devoted to Project Management Activities

The first investigation shows that the time devoted to project management tasks was in the range of 5 to 100% of the project managers' overall working time. Although we found that even those project managers who invested only between 5 and 35% of their time in leading their project regarded this to be sufficient, analysis of the data clearly indicates a correlation between the percentage of time spent for project managing and controlling and overruns in schedule and budget. As a consequence, 75% of the investigated projects were not completed within the planned schedule –

deviations from schedule were up to 150%. If more time was spent for managing the project and controlling its progress, the deviations from schedule were minor (from 20-70%). These effects can, among others, be explained by the weak progress control (e.g. no updates of the initial project plan) as a consequence of the insufficient effort associated with project management tasks.

These results could be confirmed by the second investigation. On average the project manager could only spend 25% of his overall working time on managing the project since he was assigned to up to four other software projects. According to the plan, the overall effort estimated for project management was 15 manweeks, which is less than 5% of the planned overall project effort. This project, too, was completed with enormous cost and schedule overruns. In contrast to the first study, the project manager did admit that the time spent was insufficient for leading a project of this size.

2.2.2 Insufficient Education of Software Project Managers

Analysis of the collected data of both investigations demonstrates a lack of education of project managers. While training the developers on technical topics is regarded as necessary if ability and knowledge of team members is insufficient, none of the interviewed project managers had ever participated in courses on software project management. Hence, technical experts being promoted into managerial positions had never been trained on the new skills required for successfully leading a software project.

As a consequence, project managers did not know important project management techniques. E.g., the only method that was applied by every project manager was planning of schedule and milestones by means of Gantt charts because this was supported by tools (e.g. MS Project). But project planning was done without applying any cost estimation method or systematic risk management. Analysis revealed that there was no explicit identification of risks in the project plan, and, as a consequence, no explicit management of risks. Risks were first mentioned in the status reports at the time of their appearance, but again without consequences on the plan.

In most projects there was a lack of regular and explicit updates of the project plan. We identified a clear deficit of project tracking and control.

Our conclusion is that organizations and project managers do not know how important and useful the application of basic management techniques is in order to effectively plan and control a software project. As a consequence, training on project management techniques is often neglected.

2.2.3 Important Quantitative Data were not Available for Project Management

In all investigated projects there was no systematic collection of quantitative project data that could be used to support project planning and to control the progress of the current project. Consequently, we found that the estimation of effort and time for new projects was always done based on the project managers' experience or on the experience of others. Looking at the answers we got from project managers, it is obvious that the estimations based on experiences are highly inaccurate. E.g., there

were often unrealistic estimations concerning the productivity of team members or the effort necessary for inter-project communication and quality assurance activities. The second investigation shows a dramatic underestimation of the effort needed for producing the user documentation. Because of this, all testing activities were cancelled in order to complete the user manuals. Consequently, project planning based on unrealistic estimations was one of the main reasons for the enormous time and schedule overruns.

If there are no metrics defined to measure quality and progress, project managers cannot realistically plan for further quality assurance tasks or effort necessary to complete the software product. Hence, measuring a software development project is the basis for any kind of software process improvement.

2.2.4 Quality Assurance was Often Neglected

Another aspect concerns quality assurance. Our investigations show that the importance and value of quality assurance activities was underestimated by project managers as well as by the development organization. Some project managers stated that their projects had no quality assurance function at all. In some projects, the project manager or the developers were responsible for quality assurance. In case of the project investigated in the second study, analysis of the initial project plan shows that it was planned to perform quality assurance activities such as design or code reviews. No effort, however, was assigned to these tasks. Only for software testing (as a quality assurance activity) information such as effort and time was estimated and explicitly mentioned in the project plan. As a consequence, except one formal review of the functional specification, only testing activities have been performed. But although software testing was considered necessary, it was stopped because of time pressure. This is also true for most of the projects of the first study. In case the projects ran into schedule problems, the most common way to short-cut them was to skip quality assurance activities. Hence, the schedule constraints always had priority over quality goals. This may be reasonable in some cases, but it should not be the standard case.

3 The SESAM Simulation System

Besides others, these findings clearly demonstrate a lack of training of project managers. To reduce problems in leading software projects, managers need to be trained in advance. In this chapter, we present our approach for teaching software project management by executing models of software development projects in SESAM. We will introduce a simulation model, the QA model, that aims at demonstrating important effects in software projects (especially effects concerning quality assurance).

3.1 Basic Idea

Providing a training environment for future project managers is one of the most important aspects of the SESAM project. The project aims at creating a software

system that enables users to interactively simulate software projects. A potential system user is a (future) project manager, also called student in the remainder. A user may or may not have practical experience, however, a sound theoretical software engineering and project management education is strongly recommended.

The user of the simulator receives a project description and plans the project according to his or her abilities. The student then communicates with the system using a simple interface. He or she can trigger game actions that correspond to actions within real projects, like hiring new people, or assigning tasks to staff members. Depending on the actions triggered, the student gets reactions from the system. The success of the simulated project or its failure, respectively, depends on the actions and decisions the project manager has taken. The progress of the simulated project is reflected quantitatively using various process and product attributes. After the simulation is finished, the process as well as the resulting products can be analysed. This approach offers, among others, the following opportunities:

- The simulation allows for demonstrating and explaining how the resources used, or the management approach adopted, influence the overall project results.
- The simulation is a means of examining the consequences of changing the process, or the resources.
- The simulation of software projects allows for coaching future project managers by exposing them to reality like problems and situations.
- The models can be tailored to specific development environments in order to support project planning and project control.

Quantitative modeling is not a new idea. The pioneering work of Tarek Abdel-Hamid aimed at gaining a fundamental understanding of software project management processes [8]. His results have influenced a number of similar approaches [9], [10], the basic idea of which is to describe and simulate software processes using system dynamics. The drawback of system dynamics models is that they are neither interactive, nor fine-grained. While well suited to describe quantitative aspects, they do not provide means of interaction between model and student for training purposes.

3.2 A SESAM Model – The QA Model

In the following section, a model of a software development process, the Quality Assurance model (QA model), is introduced. We will discuss the modeling approach, basic model assumptions, and the metrics used. Details on conceptual and technical aspects of the SESAM simulation system can be found in [11].

3.2.1 Educational Goals and Model Requirements

Before developing the new simulation model, the educational goals have been elaborated. We found it most important to provide sufficiently realistic simulation models so that students accept the system as a training environment. Additionally, the educational success strongly depends on the fact that the student is *not guided* by

the system but that the student is forced to control the project interactively and based on his or her own decisions. One result of our analysis was the derivation of the following basic educational goals.

- The model should be flexible so that the results of different management approaches can be compared. Students should develop their own solutions to a given management problem instead of following a predefined course of the project.
- The model should be fine-grained. The commands the simulation model accepts should resemble those actions a project manager would take in reality. Thus, experience gained during simulation can be transferred to real projects.
- The model should cover all phases of software development. Particularly in the later stages of a software project, significant deficiencies or failings can and will be unveiled. Students should be confronted with these serious problems. Only after they have experienced these difficulties, they have a chance to avoid them in real projects.

Preliminary studies within the SESAM project proved it difficult to develop a universal model that fits any particular software project independent of its size. The QA model concentrates on small to medium size projects and covers all development activities from requirements specification to product delivery. Software development can be supported by quality assurance activities. The QA model includes reviews of all documents as well as different kinds of tests. As the results of the case studies suggest, demonstrating the costs and effects of quality assurance activities is especially important for most SESAM users.

3.2.2 Model Assumptions

SESAM models contain known effects in software projects. The knowledge and the assumptions captured in the QA model mainly stem from the software engineering literature and from experts. The literature search focused on empirical studies. The results of these studies help to describe the cause and effect relationships in a quantitative way. A few model assumptions are now discussed.

(1) The productivity of the individual developer decreases as the size of the project team increases since a growing amount of time is spent for communication (e.g. in meetings). Brooks [12] demonstrated this correlation first.

(2) Developers' abilities and experiences fundamentally influence their productivity as well as the quality of the results they produce [13]. This effect is also reflected in different cost estimation methods, e.g. COCOMO [14].

(3) Reviews allow to find errors early in the development process. Completeness and correctness of the reference documents significantly determine the effectiveness of the review. Besides, reviews are influenced by other factors like developers' review experience or the effort devoted to the review preparation [15], [16].

(4) The later an error that was introduced early in the development process is uncovered, the higher are the costs to remove that error [17].

3.2.3 Modeling Concepts and Metrics

The selected dynamic effects have to be represented in the simulation model. Therefore, relevant entities have to be identified and are further described by attributes. Definition of a coherent metrics system then allows to reflect changes of the actual project state in a predominantly quantitative way.

The basic approach taken for the QA model consists in modeling documents and developers. Developers produce or examine or improve different types of software documents depending on the task assigned by the project manager. When producing software, the developers have to identify, to transform, and to document customer requirements.

Documents. The software that is produced during the simulated project is characterized by its contents, its size, and its quality. The *content* of the software is modeled by requirements. These requirements are measured using the Function Point method [18]. This approach allows to abstract from a specific project task, while preserving the opportunity to evaluate the simulated results quantitatively. The metric is sufficiently abstract for being suitable for all types of software produced during a project. Thus, a uniform approach for modeling different types of documents could be chosen.

The *size* of a document is given in number of pages or lines of code, respectively. The size is derived from the number of adjusted function points (AFP) contained in a document. Thereby, the languages or notations applied for the respective document are considered. The *quality* of a document is primarily characterized by assigning errors to the requirements contained in a document. We distinguish analysis, design, coding, and documentation errors to focus on the time of fault introduction. Modeling the documents' quality is complemented by assigning further individual quality attributes (like readability) where necessary.

Developers. Developers are the most important resource in software projects. Their knowledge and their experiences significantly influence the project results. Consequently, the description of the developers in the QA model also concentrates on their abilities. First the experiences of the developers concerning different tasks (e.g. specifying, coding, or reviewing) are rated at an ordinal scale (no/low/average/ high). Second the skills regarding various notations and languages are evaluated. Thus, a detailed individual profile of each simulated developer is constructed.

The different model elements are then dynamically combined. If, for example, a developer is asked to design the system, he or she searches for a reference document, e.g. the specification. This information is taken as a basis, and is transferred into the system design depending on the individual skills. In the QA model, this software production task is described by the following three attributes: the productivity rate (AFP/hour), the error rate (number of errors/AFP), and the loss quote (% [AFP]). Software examination and correction is modeled in a similar way. These activities, however, allow to detect and to remove errors instead of introducing new ones.

Finally, the actual development process is characterized by cost, effort, and duration.

3.2.4 Model Parameterization

A significant part of the cause and effect relationships covered in the QA model is quantitatively supported by the data provided in [6]. This data stems from observing a broad range of software projects that are further distinguished by their project class. All productivity information is expressed on a function point basis and also given for individual activities. Additionally, Jones [6] provides data material concerning errors, costs, and duration. Where necessary, other investigations have been taken into consideration, e.g. [15] or [16].

Despite this approach, there is still a number of model aspects that cannot be bolstered by empirical data. Therefore, quantification of the QA model is complemented by estimated parameters that have to be validated, as more empirical data is available. Simulation experiments, however, showed the model to produce plausible results. Results of different simulation runs are presented and discussed in [19].

4 How Project Management and Investigations can Benefit from the SESAM System and Vice Versa

As mentioned before, the results of the two industrial case studies provided valuable support when developing the QA model. Analysis of the data allowed to identify important aspects that should be focused on in the model. The results, however, were only used in a qualitative respect. Below, we give reasons for this situation.

We then analyse how empirical studies and model development have to be coordinated to gain the highest possible mutual benefit. Especially, we discuss the requirements for also using the quantitative results of empirical studies within a simulation model.

The empirical data collected in the case studies could not be used to bolster the QA model quantitatively. This has the following reasons: (1) SESAM models aim at reflecting reality as closely as possible. Comprehensive models are best quantified using investigations that cover a number of different aspects on a broad statistical basis. This, however, could not be ensured by the studies described above. (2) We could not measure all aspects covered in the QA model since one of the main deficiencies of project management is that hardly any quantitative data were available. (3) To ensure that the data material "fits" the model, the metrics defined for the data collection should be the same as those used in the model. But, since at least model development is a long term activity, the tasks did overlap. At the time the empirical studies started, the metrics system has not been fully defined.

However, even without being quantitatively bolstered, the QA model is highly suitable to call attention to the problems identified in our investigations. First of all, using the SESAM simulation system helps (future) project managers to gain experience without wasting time and money in real projects. Users can examine and compare the overall project results depending on the management approach taken. Since the QA model covers all software development activities, project managers can learn about the effort needed for different project tasks (e.g. specification, user

documentation, or code reviews). Furthermore the students are confronted with reality like problems, so that they become aware of the potential risks and learn how to manage them adequately. Hence, they will get an impression of how important it is to assess and monitor risks during the software project. They experience which quantitative information is needed to be able to assess product quality and project progress (e.g. to estimate the remaining effort or to come to decisions). Probably, this quantitative data will then be collected in real projects they have to manage. Another effect that cannot be taught by text books is to determine the adequate amount of time required for project management tasks.

Specifically, the problem of quality assurance neglection identified in both investigations is best addressed by the QA model since it emphasizes the effects of quality assurance activities. If (future) project managers interactively simulate the software project modeled by the QA model, they will experience the consequences of neglecting quality assurance (e.g. skipping testing activities because of schedule problems). They will learn how quality assurance influences the overall project results.

We will now give an overview of how these tasks should be coordinated best to gain mutual benefit. To ease planning of empirical investigations, the QA model can be used as an input for defining the metrics applied for data collection, i.e. the metric definitions can be matched to those used in the model. Vice versa, the QA model can then profit by the results of these investigations since the collected data can contribute to the validation of the QA model (e.g. the representation of dynamic effects). Hence, empirical data will be provided for those parameters that could only be estimated during model development. Figure 4.1 demonstrates how SESAM models and empirical investigations should influence each other.

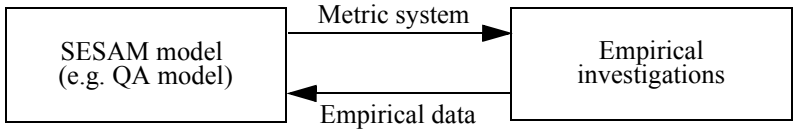

Figure 4.1: Interrelation between SESAM models and empirical investigations

The success of the suggested coordination of SESAM models and case studies mainly depends on the data available in the investigated software projects. Hence, to ensure that all aspects defined in the case study can be measured, it is first recommended to introduce a metrics program for the company in consideration based on the metrics system used in the SESAM model, e.g. the QA model. Vice versa, the collected data allow for tailoring the SESAM model to the company's environment. Adjusting the model to a specific environment is important for the following facts. (1) Transfer of the experiences gained by simulations to practical work is simplified if the model used for the simulation is able to reflect the student's reality. (2) If tailored quantitatively to a specific development environment, SESAM models can support planning and control of projects developed in this environment. (3) Furthermore it is then possible to investigate the effects of process changes before

introducing them in real software projects.

Thus, to develop realistic SESAM models we need more company specific data. Vice versa, industry benefits from models tailored to their specific needs. Figure 4.2 demonstrates this interrelation.

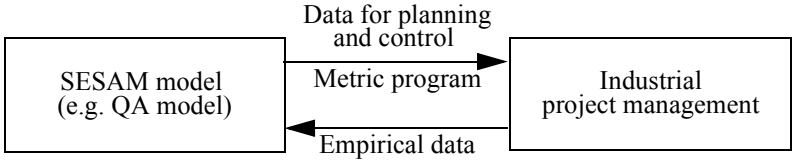

Figure 4.2: Interrelation between SESAM models and software project management

5 Conclusions and Outlook

In this paper, we have presented a new approach for teaching software project management. The basic idea is to provide reality-like experiences to (future) project managers by simulating software projects. The development of simulation models for the SESAM system highly benefits from empirical investigations. They are not only hints to key problem areas that should be considered in the simulation model. Empirical investigations do also provide quantitative data which are needed for the simulation. New empirical investigations that collect the data already used in the simulation model allow for tailoring the models to a specific development environment. In return, specific SESAM models are highly suitable for teaching project managers that work in that environment.

Implementation of a simulation model proved the approach to be suitable for project management education [20]. Since our current models are based on average industrial data, we now plan to collect specific industrial data in further empirical investigations. Then it is possible to perform specific education programs as well as to validate the simulation model in an industrial environment.

References

1. Wohlwend H, Rosenbaum S. Schlumberger‘s Software Improvement Program. IEEE Transactions on Software Engineering (20), No. 11, 1994, pp 833 - 839
2. Lichter H, Mandl-Striegnitz P. Software-Projektmanagement in der Industrie – Erfahrungen und Analysen. Bericht SL-2/96, Universität Stuttgart, 1996, in german
3. Drappa A. Erhebung von Metriken in industriellen Softwareprojekten. Diplomarbeit, Universität Stuttgart, 1993, in german
4. Jones C. Programming Productivity. McGraw-Hill, New York, 1986
5. Paulk M, Curtis B, Chrissies M, Weber C. Capability Maturity Model for Software, Version 1.1. Carnegie Mellon University, 1993

6. Jones C. Applied Software Measurement. 2nd Ed, McGraw-Hill, New York, 1996
7. Mandl-Striegnitz P, Lichter H. A Case Study on Project Management in Industry – Experiences and Conclusions. Proceedings of the European Software Measurement Conference (FESMA), 06. - 08. May 1998, Antwerp, Belgium, 1998, pp 305 - 313
8. Abdel-Hamid TK, Madnick SE. Software Project Dynamics: An Integrated Approach. Prentice Hall, Englewood Cliffs, 1991
9. Levary RR, Lin CY. Modelling the Software Development Process using an Expert Simulation System Having Fuzzy Logic. Software – Practice and Experience (21/2), 1991, pp 133 - 148
10. Madachy R. System Dynamics Modeling of an Inspection-Based Process. Proceedings of the 18th International Conference on Software Engineering, 25. - 29. March 1996, Berlin (Germany), 1996, pp 376 - 386
11. Melchisedech R, Deininger M, Drappa A, et al. SESAM – A Software Engineering Education Tool Based on Graph Grammars. Bulletin of the European Association for Theoretical Computer Science (EATCS), No. 58, 1996, pp 198 - 221
12. Brooks FP. The Mythical Man-Month. Datamation, 1974, pp 44 - 52.
13. Sackman H, et al. Exploratory Experimental Studies Comparing Online and Offline Programming Performance. Communications of the ACM (11), No. 1, 1968
14. Boehm BW. Software Engineering Economics. Prentice Hall, Englewood Cliffs, 1981
15. Weller EF. Lessons from Three Years of Inspection Data. IEEE Software (10), No. 5, 1993, pp 38 - 45
16. Porter AA, Siy HP, Toman CA, et al. An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development. IEEE Transactions on Software Engineering (23), No. 6, 1997, pp 329 - 346
17. Ludewig J, Frühauf K, Sandmayr H. Software-Prüfung. Eine Anleitung zum Test und zur Inspektion. 2. Auflage, vdf Hochschulverlag, Zürich, 1995, in german
18. International Function Point Users Group (IFPUG). Function Point Counting Practices Manual, Release 4.0, 1994
19. Drappa A, Ludewig J. Quantitative Modeling for the Interactive Simulation of Software Projects. Proceedings of the Software Process Simulation Modeling Workshop ProSim 98, 22. - 24. June 1998, Silver Falls, OR (USA), 1998
20. Deininger M, Schneider K. Teaching Software Project Management by Simulation – Experiences with a Comprehensive Model. In: Díaz-Herrera, J. L. (Hrsg.): Software Engineering Education. Proceedings of the 7th Conference on Software Engineering Education, January 1994, San Antonio, Texas, USA. Springer (Berlin), 1994, pp 227 - 242