IT & C Projects Duration Assessment Based on Audit and Software Reengineering

Cosmin TOMOZEI, University of Bacău
Marius VETRICI, Cristian AMANCEI, Academy of Economic Studies Bucharest
cosmin.tomozei@ub.ro, mariusvetrici@softmentor.ro, cristian.amancei@ie.ase.ro

This paper analyses the effect of applying the core elements of software engineering and reengineering, probabilistic simulations and system development auditing to software development projects. Our main focus is reducing software development project duration. Due to the fast changing economy, the need for efficiency and productivity is greater than ever. Optimal allocation of resources has proved to be the main element contributing to an increase in efficiency.

JEL Classification: L86 Computer Software, O22 Project Analysis
Keywords: *Reengineering, audit, project duration assessment, Monte Carlo simulation.*

1 Introduction

The grand majority of software development projects are known to be delayed and over the budget. Most of them hit schedule and budget overruns of 25% to 100% and sometimes even more [1], [2], [3], [4].

The prerequisite for defining an accurate project delivery date is a precise estimation of the project duration. Existing models are rather imprecise because the forecast value is to a certain extent distant from the real one. The large discrepancies between the estimated duration and the actual schedule of an ongoing project prematurely ended it in order to prevent further damages and losses. The [2] research reveals that only one project in three is considered successful, whereas one project in five is a total disaster. Taking this into account, it is imperative to look for new software project duration forecast models that will be able to outspring results that are more realistic.

The aim of this research is to bridge the gap between the forecasted software project duration and the actual project duration. Hence we go into great depth with analyzing the existing project duration assessment models and for each one we stress on its advantages and disadvantages. Then, we present software reengineering as a means of delivering timely, high performance software projects. Samples of students' projects are analyzed in terms of size and duration. Last but not least, the benefits of an independent auditor certification regarding project quality are presented in great detail.

2. The taxonomy of duration assessment models

The range of duration assessment techniques and methods significantly broadened its coverage in the last years so that now we have sophisticated mathematical and statistical models and even expert system based estimation models. Figure 1 depicts the classification of existing models:

Expertise-based methods rely on the subjective judgment of a human expert or a group of experts and are the most widely used methods for project duration estimation [5]. Unlike rigorous estimation methods, these methods rely on the personal intuition and on the experience gained by the human expert in question [6]. For example, according to Delphi method, a panel of experts is required to make an estimation regarding a project. After the first step, the estimations are debated and then the experts go to a second stage of estimations.

After each estimation stage, some elements and some details will be left out, while others will be greatly emphasized. The process is iteratively repeated until a common agreed duration is reached [8].

Learning-oriented techniques try to identify a similar software development project and infer the duration out of the past experiences and the differences between the old and the

new project [9]. The advantage of this class of techniques over the expertise-based ones is that in this case estimations are grounded on real life facts and on palpable examples and not on the general experience of the experts. The disadvantage of these techniques is the fact that it is not very obvious how the two projects should be compared, what the key variables that should be tracked are and what the issues that should be left out are. The identification of the key variables is a tedious, time-consuming task because of the very particular nature of software projects.

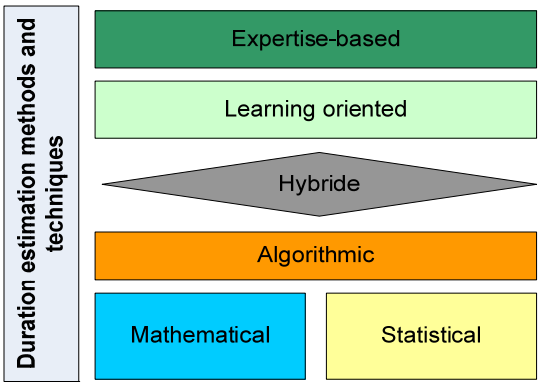

Fig. 1. The classification of duration assessment methods and techniques for software projects [7]

The algorithmic methods use iterative approaches based on mathematical formulae. They take as input data the size of the software project (counted in function points or lines of source code) and parameters like hardware and software development platform, team experience, manager experience and the employed development methodology. Based on the input data the algorithm assesses project duration together with an index of the estimation accuracy. The algorithmic methods are iteratively run several times in order to refine the input parameters' values and to enhance the estimation accuracy. The limitation of this class of methods comes out when the algorithm is fed with uncalibrated or not validated data. Most of the algorithmic methods offer estimations for duration, for effort and even for the total cost of the projects. Among them are COCOMO and COCOMO 2.0 [10], SLIM, Neural Networks, Critical Path Method, Critical Chain Method, PERT.

Mathematical-statistical models are particularly useful when confronted with large sets of historical data available for analysis. Such models include the linear regression and the multiple regression [11] advances a new method for project duration estimation that takes into account the time consumed with inter task communication.

Table 1. Duration assessment methods for software projects

Name	**Pros**	**Cons**
Expertise-based methods	These are the most flexible methods that can be easily adapted from project to project in order to enhance the quality of the estimations of the amount of time needed.	Too subjective. Depend on the experience of the experts in question.
Learning-oriented techniques	Are based on real life examples that have been previously executed.	The necessity to identify the key-variables is a daunting, time-consuming task because of the specific aspects of every project.
Algorithmic methods	-Are able to refine their estimates on subsequent iterative algorithm execution. -Can be easily adapted to the variations of the input values.	The estimations can have a very low quality when the input data has not been properly validated and calibrated.
Mathematical-statistical models	- Are easy to develop. - Have a very good academic background.	Need a large set of historical data.
Hybrid methods	Are the most efficient since they combine key aspects from all other methods.	Are immature, undeveloped and lack solid formalization.

Hybrid methods have been created in order to overcome the increasing uncertainty and complexity of software projects. This class of methods combines algorithmic, statistical, mathematical and expertise-based methods into a single unitary method. An example of such a method is the Metrix model [12]. This is a stochastic model that addresses the project duration uncertainty by running Monte Carlo simulations over the activity graph. The advantage of this approach is that the model produces an interval for the possible project durations and a probability distribution. Thus, one is able to know the possible project durations together with the probability that certain duration will materialize. Table 1 generalizes existing duration estimation methods for software projects together with their pros and cons.

3. Software reengineering as a key process in project deadline fitting

One of the most important components of the large-scale application development apparatus is software reengineering. The duration of project development, in this case is relatively easily determined when the methodologies, methods and techniques of reengineering are applied by the development company.

Let's take into consideration a modular application in which every model has a certain level of autonomy. This application will be subjected to a formal analysis, consisting of the amount of software metrics that will describe by means of indicators its behavior in some problematic situations. Before going into analysis, we will briefly depict the probability function we use. In [13] the probability space is defined as the triplet (Ω, F, P) where Ω is a nonempty set meaning the sample space, each element ω of Ω is called outcome and F is a set of subsets of Ω called events. The characteristics of the probability function P are described by the following axioms [13]:

$$P[A] \geq 0 \quad for \quad every \quad A \in F \quad (1)$$

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{n} P(A_i) \quad (2)$$

$$P[\Omega] = 1 \quad (3)$$

Preliminary probabilistic calculations are made in order to estimate how much time it would take to the project elements to get transformed, implemented and tested in order to satisfy the new demands of the client. Random probability vectors are to be used in order to describe each component or module of the project. If the project has n autonomous components, the behavior of the project may be formalized using a random probability vector $\boldsymbol{X}$ of dimension $\boldsymbol{n}$. The expectation of a random variable is well known as the following formalism [13]. The expectation may be formally described by a sum or integral.

$$E[X] = \int_{\Omega} X(\omega) P d[\omega] \quad (4)$$

$$E[X] = \sum_i X_i P\{X = x_i\} \quad (5)$$

The expectation regarding the moment of the projects finalization as well as the expectation for the objectives to be accomplished with minimum errors has to be related with the expectance of every component of the software entity. Tables of association are used in order to provide a graphical view of the correspondence between each functional module and the expectation associated to it. Software reengineering does minimize the time for each module and maximizes the probability of success. In this case, the following formulae present in an inequality the optimizations of expectations for each entity.

$$T = \sum_{i=1}^{n} t_i \quad (6)$$

$$Min(T) = Min\left(\sum_{i=1}^{n} t_i\right) \quad (7)$$

$$E[X] = \sum_i X_i P * y_i <= Min\left(\sum_{i=1}^{n} t_i\right) \quad (8)$$

T represents the total amount of time needed to complete the entire software application, and appears like a sum of the amounts of time needed for each functional module. If the applications source code is developed in

an object oriented programming language, a module is being associated with a class. Every class has also components whose necessary amount of time may be measured. If the management decides to increase the level of precision of (8), it is very simple, just by considering other indices, such as *j*.

$$\sum_i y_i = Dev\left(\sum_i t_i * x_i\right) \quad (9)$$

The development function *Dev* [15] is the function that brings together as arguments the functional modules and the chronological allocation corresponding to them. The outputs are the results consisting of the software's functionality. Software reengineering maximizes the probability of time reduction between the decision of realization and the moment of delivery. As a result, the development cycle becomes shorter and better. Life cycle management implies that both software maintenance policies and security policies deal with probabilistic estimations and simulations, such as *Monte Carlo Simulation*.

Correlation and covariance provide a clear image about the relation and determination between entities. In [13] the correlation between entities X and Y, $E[X,Y]$ and the covariance $Cov[X,Y]$, where X and Y are random variables are:

$$E[X,Y] = E[X] * E[Y] \quad (10)$$

$$Cov[X,Y] = E[(X - E[x])E(Y - E[Y])] \quad (11)$$

The process of programming must be economically efficient. Decreasing of costs is also important, as a key factor in efficiency calculations. Expectation of costs is also calculated as a mean, like in a random process case. In modular applications, expectation of the total cost is a sum of expectations for every particular module. The total cost function appears like a function depending on the prices or costs of every component of the vector of modules.

Let's consider the cost function as $CT(w,y) = \min_x V * x$ which may also be described as $CT(w_1...w_n, y) = \sum_{i=1}^{n} w_i x_i(w_1..w_n; y)$ where w_i represents the amount of financial resources spent for the unit *i*, y the level of output and represents the level of the demand for the unit *i*. The target cost appears in this case as a level of expectation. As a consequence, $E[W] = (E[w_1],...,E[w_n])$ and $E[CT(w_1,...,w_n; y)] =$ $CT(E[w_1],...,E[w_n]; y) =$ $\sum_{i=1}^{n} E[w_i]x_i(E[w_1]..E[w_n]; y)$ because the expectation of a sum of non negative variables is the sum of expectations.

As a conclusion we can state that software reengineering process results in a reduction of the time needed to deliver a project. The cost function, which is optimized by minimization naturally, and the fixing of deadlines are key components in increasing efficiency.

4. IT&C project sample analysis

We will put together the following analysis on samples of projects in order to offer a reliable image of how the time resources allocation is optimized, concerning the duration of the development. The above formulae are being implemented and experimental results are determined.

In Table 2 the specifications and metrics for 35 small software projects are described. The projects were developed by the students from the Computer Science and Accounting Departments at Bacau State University. Projects have been developed by teams ranging in size from 1 to three members. The teams had to measure the time spent for project development, the lines of code written and the number of methods used. The projects have been evaluated by two teachers, each one of them giving a mark, the final mark appearing as an average of the former marks. As a preliminary conclusion, we state that the best

projects have been very well structured, each method having between 20 and 30 lines of source code. The average of durations for the best projects is 59 hours of working and standard deviation of 38.09 hours. For the same projects, the number of executable lines of source code is 918, with a standard deviation of 637 lines.

We found out there is a correlation coefficient of 0.79 between the number of methods and the number of code lines, which indicates that the increasing number of lines is also increasing the number of methods, which is obvious if the projects are well structured (see figure 2). These estimations allow us to build precise demands for students or development teams in companies and forecast the obtained results. For example, for a small project of 59 hours of code elaboration, without considering the part of user interface or web design, but with testing and debugging included, the number of code lines is around 1000. This number of source code lines can increase or decrease, depending on the experience of the developer and the complexity of the demand. For programs that require advanced graphics, such as gaming and animation the duration increases significantly in comparison with other types of projects.

Table 2. Students Projects Sample. Description and Evaluation

Nr.	Spec	Nr. pers.	Title	Nr. methods	Nr. lines	Duration (h)	Methods per hour	Lines per hour	Lines per method	Grade
1	Acc	2	Pers. Loan	10	1400	15	0.67	93.34	140	9
2	Inf	1	Reception note	17	128	20	0.85	6.4	7.53	6
3	Inf	1	Black Jack	63	1397	15	4.2	93.14	22.18	10
4	Inf	2	Painter	50	1000	75	0.67	13.34	20	10
5	Inf	1	Shooter	13	150	15	0.87	10	11.54	8
6	Inf	1	Tic-Tac- Toe	24	500	50	0.48	10	20.84	5
7	Acc	3	Minesweeper	7	225	15	0.47	15	32.15	8
8	Inf	1	Undetermined	14	330	150	0.1	2.2	23.58	7
9	Acc	2	Statistical functions	22	233	50	0.44	4.66	10.6	7
10	Acc	2	Auto test	275	2930	50	5.5	58.6	10.66	9
11	Acc	2	Accept or not	54	1170	42	1.29	27.86	21.67	7
12	Acc	1	Depreciation	25	296	75	0.34	3.95	11.84	9
13	Acc	1	Depreciation	45	726	96	0.47	7.57	16.14	9
14	Acc	1	Population	11	133	35	0.32	3.8	12.1	7.5
15	Acc	2	Payment	15	250	40	0.38	6.25	16.67	8.5
16	Acc	2	Orders	10	100	10	1	10	10	8
17	Acc	1	Test Choice	4	132	48	0.09	2.75	33	5
18	Acc	2	Annuities	43	300	40	1.08	7.5	6.98	8
19	Acc	1	Delivery	3	65	5	0.6	13	21.67	7
20	Acc	2	Payment	13	110	20	0.65	5.5	8.47	7
21	Acc	1	Sudoku	5	70	7	0.72	10	14	5
22	Acc	1	Mp3 Player	52	475	120	0.44	3.96	9.14	10
23	Acc	2	Play memory	3	150	30	0.1	5	50	5
24	Acc	1	Exchange	2	29	4	0.5	7.25	14.5	5
25	Acc	2	Payment	7	200	24	0.3	8.34	28.58	6
26	Acc	2	Payment	10	176	20	0.5	8.8	17.6	5
27	Acc	2	Ball Game	9	160	9	1	17.78	17.78	5
28	Acc	2	Annuities	5	103	30	0.17	3.44	20.6	6
29	Acc	2	Test Person	5	81	30	0.17	2.7	16.2	8
30	Acc	1	Leasing	140	3	8	17.5	0.38	0.03	10
31	Acc	1	Depreciation	30	900	15	2	60	30	10
32	Acc	2	Consolidation	100	2000	70	1.43	28.58	20	10
33	Acc	2	Zodiac	7	800	30	0.24	26.67	114.29	8
34	Acc	2	Galloway	35	400	75	0.47	5.34	11.43	8
35	Acc	2	Encyclopedia	15	200	72	0.21	2.78	13.34	7.5

Table 2 presents the experimental data and results from the student's projects evaluation. On every step of the development, they have been guided to make their work reliable. We have considered as being ready for delivery the projects which accomplished at least 90% of the initial objectives.

During the development process, new requirements have been added so the projects suffered major transformations. Reengineering proved to be a way of increasing the efficiency and we have noticed significant decreasing of the development time. For example, the 30th project, a program that used to compute the coefficient of allowance for depreciation have been easily transformed into bills payable book for the leasing, with a very small effort. By reusing the existing amount of methods, which were generally and reliable implemented.

Optimization of time spending proves to be a key factor in efficient allocations of resources and reengineering contributes to it.

Figure 2 illustrates the correlation between the number of source code lines and the number of methods after removing outlier values.

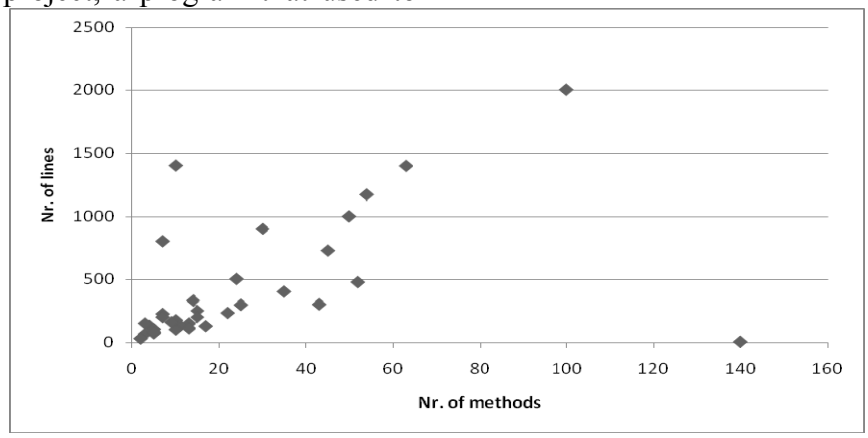

Fig. 2. Correlation between the number of lines and the number of methods

5. Time resource allocation optimization

The fast changing economic environment, competition and the dynamics of globalization are the main challengers for software development project efficiency. Fast changing economic environments and the preservation of resources (e.g. time, energy, and money) strengthen the idea of optimal allocation in every economic activity in order to be efficient.

In software industry, due to the competitive environment and to the need of companies to "strive forward to survive" [14], it is compulsory to reduce the development time. Reduction of development time means that the same objectives, which had been previously achieved by applying the steps of the development cycle in a specific time, have to be applied with some modifications in a shorter period. *Software reengineering* and especially *semantic reengineering* [15] are to become key factors in reusing the existing amount of software modules and functionalities to achieve the new goals, just by making updates and modifications.

In [16] it is stated that one way of growing efficiency in software development is by reducing complexity and product size, by getting a higher level of abstraction and by using visual modeling notations for reducing the human generated source material. We believe that to be just a partial approach, human factor being the most important constituent of software companies.

For each step of the development cycle an explicit amount of time is dedicated. Firstly, the management of the project talks to the beneficiary in order to get a clear image about the expectation that the costumer has for the ordered software.

Customers have only abstract ideas about what they want, but not exactly about what the software must do. After several discussions and explanations, the software developer will have a clear idea about the requirements of the customer and the customer

gets informed about the complexity and difficulty of his demands.
Incorrect requirements will cause unwanted modifications and consequently delays. The time for the planning and requirements identification has to be sufficient, because misunderstandings and left behinds may induce severe problems in the following steps.

Each of the following steps of software engineering will have in correspondence an amount of time in the development process. The main idea is to assign appropriate length of time, and not to waste it, because if the time resources are not efficiently allotted, the project will be delayed and consequently will be more expensive.

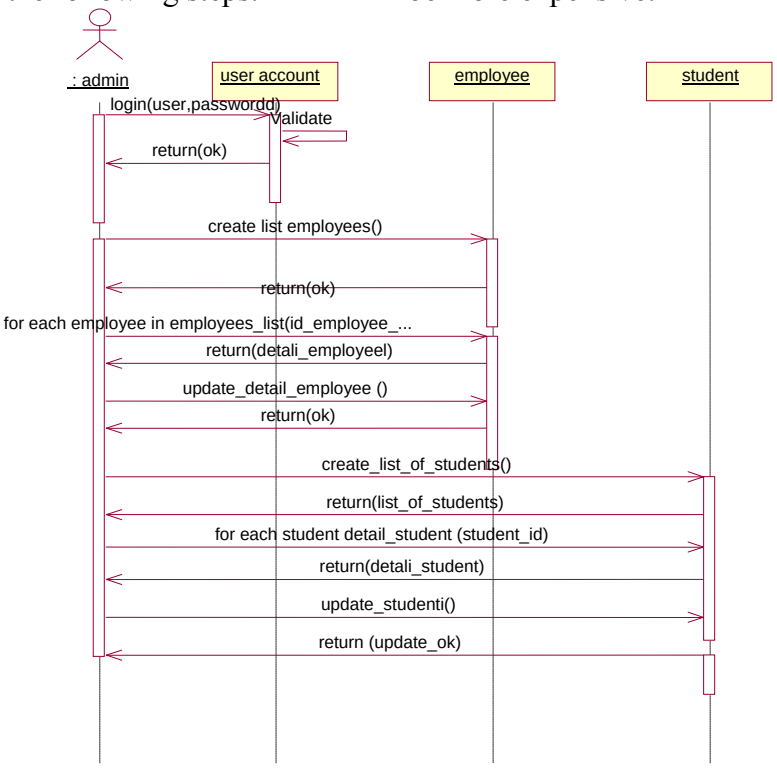

Fig. 3. Management of user accounts

Using UML diagrams for capturing the design of the software system, including the *architecture,* will save time and money. In the next stage, decisions have to be taken, regarding the number of the workforce involved in the project, the tasks each member of the team has, and to the decision to reuse existing components or to start all from the very beginning. By using preexistent middleware and automatically generated components, the project will have chance to be finished before the deadline. It is important to have both static and dynamic view of the software system, as well as the outputs, the final results. Class diagrams, sequence diagrams and use – case views consent the capturing of the existing software system design. Reengineering becomes easier to implement new functionalities and achieve new objectives of quantitative or qualitative manner.

Figure 3 is a sequence diagram of the administration module in a web application which manages the categories of users in the academic area.

6. System Development Auditing

Auditors are responsible for providing an independent, objective appraisal activity for the purpose of advising and assisting the management, staff and board of directors in the achievement of the organization's goals and objectives [17].
Auditor involvement in computer system development projects primarily focuses on providing independent assessments on whether appropriate controls incorporated in the systems and suitable project controls are employed. During the work the auditor, would:
• provide advice and assistance to project teams in the management of various risks and

controls in the systems;
- provide timely recommendations for any identified control weaknesses;
- provide suggestions on possible operational improvements based on observations, previous audit work, contacts with other audit professionals.

For system development reviews the auditor performs the following [18]:
- make an understanding of how the project will be managed and the system and business processes. Auditor review of procedures will also be discussed / confirmed with the project team;
- review the assessment of risk and the adequacy of controls to mitigate significant risks;
- review the deliverables at major stages in the project's life cycle to help evaluate risks and controls;
- perform certain tests at the appropriate stages in the systems development activities to ensure that controls are in place that mitigate significant risks in a cost-effective manner;
- provide advice and recommendations to the project team to help them meet project and organizational objectives.

Throughout all stages of the project, the auditor will review the evolving project plans and perform general assessment of project controls.

Information System auditors typically execute three types of reviews of the systems development process: a pre-implementation review, a parallel review and a post implementation review. During a pre-implementation review, the information system auditor investigates the proposed methodology and considers its applicability and the potential risks associated with the systems development project. In a parallel review, the information systems auditor reviews the pertinent stages of the methodology as they proceed and, subsequently, calls attention to possible risks and provides suitable risk mitigation approaches.

Finally, during a post implementation review, the information system auditor reviews the relevant stages of the methodology after the systems development project has been completed.

Regardless of the organization size and the services it provides, internal auditors should look for the symptoms that show systemic problems with the IT department project management approach. Usually, the presence of some or all the symptoms below can signal that something is wrong [19]:
- business users are unhappy with the quality and timeliness of the IT project's delivery;
- the IT department blames users for not stating their requirements and expectations clearly;
- the organization experiences frequent time and cost overruns on critical projects;
- questions related to project criticality elicit different responses from the IT department and senior managers;
- specific business opportunities are lost because IT systems or applications are not able to meet their goals;
- there is a low level of awareness and slow adoption of IT best practices.

At a minimum, key areas to be reviewed for any IT project include: user requirements, prioritization and scheduling, planning, resource management, training, monitoring and tracking, risk management, quality control checks, and delivery.

From the auditing point of view the following should be implemented by the organization:
- globally accepted best practices are used by the organization to manage IT projects effectively;
- complete segregated accesses on live, development and test environments are enabled;
- the visibility of the documentation for release into live is enabled;
- user acceptance testing is performed by using real test case scenarios and dummy data;
- change management software implemented in order to keep track of the developers activities;
- central repository for change requests with functions of workflow authorizations built in it.

If a software application has been subjected

to audit, a certain level of quality have already been certified. Reengineering will preserve the certified level of quality for the procedures, methods or functions implemented before and will also improve the application capabilities.

Auditing proved to be an important and problem solving approach from the project management perspective. Verifying and certifying the proportion in which software applications achieve their goals, the correspondence between the objectives and the results, as well as the significance of errors addresses the project management activity. Considering that, we may conclude that for developing high quality IT projects in a reasonable time, it is necessary to bring into play software reengineering and audit.

7. Conclusions

Concerning systematic approaches to software project duration assessment, the hybrid models and particularly the Metrix model's main advantage is that it produces a probability distribution of the software project duration and not single point estimation.

Software reengineering proves to be as well an important element for achieving a high level of efficiency and consequently, the time between the decision of doing a project and the moment of finalization is significantly shortened. For each step of the software engineering process, the amount of resources is optimally distributed in order for the project to be finished in time.

The entire project should benefit from the certification offered by an independent auditor, concerning the identified risks and the procedures which have to be implemented in order to mitigate the risks and fulfill the initial objectives.

We found out there is a correlation coefficient of 0.79 between the number of methods and the number of code lines for the project sample under analysis, which indicates that the increase in the number of lines results in an increase in the number of methods.

References

[1] A. W. Chow, B. D. Goodman, J. W. Rooney and C. D. Wyble, *Engaging a corporate community to manage technology and embrace innovation*, IBM Systems Journal, vol. 46, no. 4, 2007, pp. 639-650.

[2] ***, The Chaos Report of IT Project Failure, Standish Group, 2006

[3] ***, ERP Software Implementation Success Rates, Robbins-Gioia 2001

[4] S. McConnell, *Rapid Development*, Microsoft Press, Washington, 1996

[5] M. Jørgensen, *A review of studies on expert estimation of software development effort*, The Journal of Systems and Software, vol. 70, no. 1-2, February 2004, pp. 37-60.

[6] J., Callahan and B. Moretton *Reducing software product development time*, International Journal of Project Management vol. 19, no. 1, January 2001, pp. 1-36.

[7] V. Temnenco, Software Estimation, Enterprise-Wide, IBM The Rational Edge, vol. June 2007.

[8] C. Wiegers, *Stop Promising Miracles*, Software Development Magazine, February 2000, http://www.ddj.com/architect/184414570.

[9] C. Sungbin, *An exploratory project expert system for eliciting correlation coefficient and sequential updating of duration estimation*, Expert Systems with Applications, vol. 30, no. 4, May 2006, pp. 553-560.

[10] B. W. Boehm et al., *Software Cost Estimation with COCOMO II*, Prentice Hall, 2000.

[11] J. Uma Maheswar and K. Varghese, *Project Scheduling using Dependency Structure Matrix*, International Journal of Project Management, vol. 23, no. 3, April 2005, pp. 1042 – 1046.

[12] M. Vetrici, I. Cristian, *Knowledge Based Project Duration Estimation for Workflow Based Document Management Software Projects*, Proceeding of Knowledge Management Conference, Bucharest 2008.

[13] B. Hajek, *An Exploration of Random Processes for Engineers,* IFP Group at

University of Illinois at Urbana-Champaign 2008, pp. 1- 303.
[14] B. Johansson, J. Johnsson and Jon Bagiu, *Modular Assessment Systems Simulation,* International Precision Assembly Seminar, Chalmers Publication Library, 2004, pp. 215 – 222, http:// publications.lib.chalmers.se/cpl/record/index.xsql?pubid=7873
[15] C. Tomozei, *Hypertext Entities Semantic Web-Oriented Reengineering*, Journal of Applied Quantitative Methods, vol. III, no. 1, 2008, pp. 9- 19, www.jaqm.ro.
[16] W. Royce, *Improving Software Development Economics,* The Rational Edge, Rational Software 2001, Part I, pp. 1-7, http://www.ibm.com/developerworks /rational/library/content/RationalEdge/archives/apr01.html.
[17] T. Surcel and C. Amancei, *The IT Audit – A Major Requirement for the Quality Management and Success in the European Business Context*, The International Scientific Conference, Oradea 2008.
[18] Singleton, Tommie, *Systems Development Life Cycle and IT Audits*, Information Systems Control Journal, vol. 3, 2004, pp. 24 – 26
[19] J. Laurent and P. Leeson, *Auditing CMMI Maturity and Sarbanes-Oxley Compliance*, Information \Systems Control Journal, vol. 3, 2007, http: //www.isaca.org/AMTemplate.cfm?Section=20075&Template=/ContentManagement/ContentDisplay.cfm&ContentID=42530

Cosmin TOMOZEI is assistant lecturer at Mathematics and Computer Science Department from Faculty of Sciences of the University of Bacau. He is a PhD candidate from October 2007 at Economic Informatics Department from University of Economics, Bucharest. He holds a Master in Science - Databases- Business Support from University of Economics, Bucharest. He graduated in Economic Informatics at Faculty of Economic Cybernetics, Statistics and Informatics in 2006. His main research areas are: object oriented programming, functional programming in Lisp and F#, software reengineering and distributed applications development.

Marius VETRICI is an independent researcher in the field of software project management. He is a PhD candidate since October 2007 at Economic Informatics Department from Bucharest Academy of Economic Studies. Mr. Vetrici holds an MSc degree in Project Management from Bucharest Academy of Economic Studies and a BSc degree in Economic Informatics from the Cybernetics, Statistics and Economic Informatics from the same university. His main research areas are: software project management and document management.

Cristian AMANCEI is assistant lecturer at Academy of Economics Studies Bucharest, Faculty of Economic Cybernetics, Statistics and Informatics. He is a PhD candidate from October 2007 at Economic Informatics Department from Academy of Economic Studies. He holds a Master in Science – Computerized Project Management from Academy of Economic Studies, Bucharest. He is Certified Information Systems Auditor (CISA). He graduated in Economic Informatics at Faculty of Economic Cybernetics, Statistics and Informatics in 2006. His main research areas are: information system audit, data structures, metrics in information systems and object oriented programming.