Schedule Risk Management for Business M-Applications Development Projects

PAUL POCATILU, MARIUS VETRICI
Economic Informatics Department
Academy of Economic Studies
6 Piata Romana, Sector 1, Bucharest
ROMANIA
ppaul@ase.ro, mariusvetrici@softmentor.ro

Abstract: The grand majority of software development projects are known to be late and over the budget. Several surveys performed during the last 15 years expose a relatively poor performance in delivering successful software projects. Most of the projects hit schedule and budget overruns of 25% to 100% and sometimes even more. Even though m-applications development is a new software development field, still this type of projects is not secured against the common flaws of software development projects. Therefore, the main goal of this paper is to reduce the gap between the estimated duration of the m-application development project and the actual elapsed time. We find that legacy and proven best practices project management techniques can be successfully employed for schedule risk management. Furthermore, we present three proven software project management techniques that were successfully adapted to the development of m-applications. The first one is the estimation of m-application project duration using top-down and bottom-up approaches. The second one is the use of a set of performance metrics for project quality assessment. And the last one is the Extended Metrix model, a stochastic project duration estimation model with schedule risk analysis elements.

Key-Words: mobile applications, software development, project duration, schedule risk management, Monte Carlo simulation.

1 Introduction

The grand majority of software development projects are known to be late and over the budget. A wide range of surveys performed during the last 15 years reveal the dramatic reality of the software projects development. Most of the projects hit schedule and budget overruns of 25% to 100% and sometimes even more [1], [2], [3]. The adoption of Agile and Iterative development methods provided us with a raise in project success rate from one project in three [2] to two successful projects from three [4]. Still, there is place for improvement.

The total number of mobile phone subscribers in the world was estimated at 2.14 billion in 2005 [5], 3.3 billion in 2007 [6] and the figure is expected to increase to 90% by the year 2010. The numbers are even more impressive if we look at the mobile phone penetration rates, the highest from Asia being in Hong Kong with 1.4 mobile phones per person [7] and in Europe, Luxembourg, Lithuania and Italy hitting as high as 150 mobile phone subscriptions per 100 people [8]. Given the circumstances, m-application software development is and will be an emerging field of the software industry.

An m-application is a special type of software application particularly designed to be used on mobile processing units with limited processing power, storage memory and input capabilities such as mobile phones, smartphones, PDAs, navigation assistants, mobile guides, etc. Even though m-application development is a relatively new domain, legacy project management techniques can be successfully applied for delivering high performance in the new field.

This paper will be focusing on timely delivery of the project as a measure of project success. The prerequisite for defining an accurate project delivery date is a precise estimation of the project duration. Existing estimation models are rather imprecise because the forecasted value is to a certain extent distant from the real one. The large divergence between the estimated duration and the actual schedule of the ongoing projects prematurely ended them in order to prevent further damages and losses. Given this, it is imperative to look for new methods that will aid software project managers in forecasting and controlling project duration and, hence, the project quality.

The aim of this research is to bridge the gap between the forecasted m-application development

project duration and the actual project duration. Project duration estimation is of utmost importance for all project stakeholders. More specifically, the duration of the project is needed before the project has started. This is because other important estimations are grounded on the former metric. For example, no investor will go with a given project unless the delivery date is clearly agreed upon and a commitment has been entered into. Further to this, we go into great detail about the importance of project duration estimation, the difficulties of estimating duration and the existing duration estimation techniques.

Therefore, this paper analyses the specifics of the m-applications development projects and presents three proven software project management techniques that can be successfully adapted for timely development of m-applications. The first technique presents legacy top-down and bottom-up techniques for project duration estimation. The second project management technique relies on the use of specific m-applications development performance metrics. The performance metrics are based on customer satisfaction, the degree of objective completion and the cost of the resources involved. The third technique employs the use of Extended Metric Model, a Monte Carlo based project network simulation specifically adapted to the development of m-application. This approach takes advantage of the specifics of the m-application development environment and offers enhanced project schedule risk estimation and control.

2 Business M-applications

Business m-application development is similar to the development of personal computer applications, but there are also differences that influence the way the project is managed. Depending on the application type, m-applications development projects include not only the mobile device software, but also the other components of the system (application server, database, content management etc.).

With respect to data processing, m-applications can be divided into standalone applications and distributed applications.

Standalone mobile applications are designed to perform specific tasks without the need of a network connection. Mostly mobile applications made for PDAs are such examples of stand-alone applications.

Every operating system (Windows Mobile, Symbian) exposes specific APIs with varying degrees of complexity and architectures which are more or less well documented. In order to increase the development productivity, higher level classes libraries were developed on top of system's APIs. Usually, every library comes with a specific run-time environment.

Distributed m-applications instead need a network connection in order to operate. This type of applications may rely upon a permanent or a temporary connection. WAP (Wireless Access Protocol) based applications for mobile phones that connect to a server via Internet are an example of distributed applications. The most used distributed applications are Web-based. Figure 1 depicts the architecture of such an application.

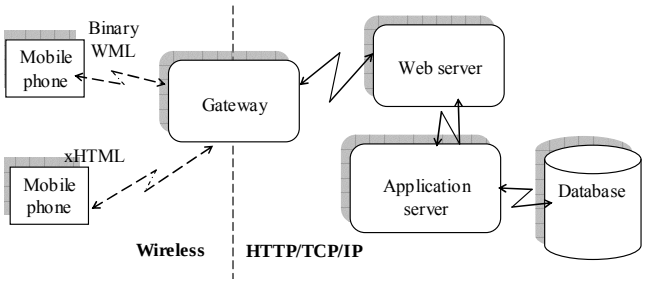

Fig. 1. Mobile Web applications architecture

The request from the WAP enabled phone is sent to the WAP gateway that makes the conversion from the WAP stack (for WAP 1.0) or from the optimized wireless or optimized HTTP/TCP/IP (WAP 2.0) to the HTTP/TCP/IP stack and encodes the network packets that will further be sent to the Web server as an HTTP request [9]. The request is processed by the Web server, and then a response is send back to the mobile phone browser through the WAP gateway that decodes the packets.

Table 1. M-application types comparison [10]

App. Type / Features	**User Interface**	**Memory**	**Processing power**	**Complexity**
Network access	Limited	High	Medium / High	High
Stand-alone	Limited	Medium / High	Medium / High	Medium / High
Web-based	Web-based	Medium	Low /Medium	Low
Database access	Limited	High	Medium / High	High

The type of application has an important influence on the size and the complexity of the m-application development project:

As it can be seen from the table 1, mobile applications that require network access and those that use databases usually have a higher complexity. This is rather obvious because this type of m-applications will have greater complexity, more classes and chances are that the demand for specific knowledge will be higher.

The size, complexity and productivity are influenced by the application's operating system. Using Java ME technology there is a high degree of portability between operating systems, but there are device specific influences.

The use of native APIs to write applications requires more effort, and the size of application (expressed as KLOC) is higher than using classes libraries.

Most of the development process is made using device software emulators that run on personal computers. Still there are differences between real-life devices and emulators. That's why there is an additional effort in testing the application even after it is considered done on the emulator.

3 Risk Factors in Business M-applications Development Process

Numerous risk factors influence the development process of business m-applications in terms of duration, costs and quality. Among them we count software development process related risk factors and m-applications development specific risk factors.

Usually, business applications allow the following functions to be used on mobile devices:

- Data input;
- Data updates;
- Data processing;
- Data verification;
- Data gathering using specific devices (camera, RFID, IO cards etc.)
- Small reports and charts;
- Database synchronization.

Public institutions instead have more specific requirements [11] which need a more thorough approach and should be implemented on mobile devices accordingly. The m-learning applications have their own software characteristics [12].

Business m-applications development process, like all projects, needs to be on time, on budget and on scope within a quality level. Most of the projects unfortunately are only able to satisfy two out of three constraints.

The risks identified from past projects were centralized and several risk classes were identified. The risk factors are related to: people, process, infrastructure (hardware and software) and to external environment, figure 2.

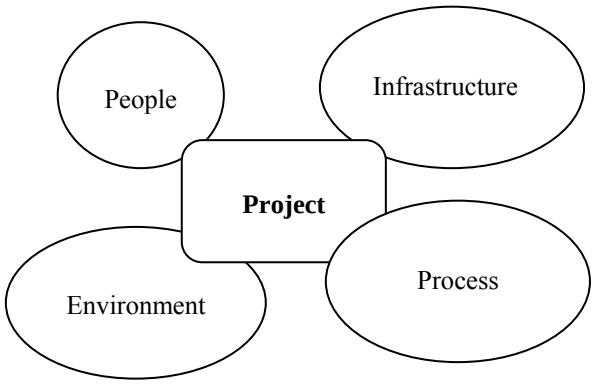

Fig. 2. Risk Factors

People related risk factors are:

- The lack of experience;
- People education;
- Knowledge in business field and m-applications development;
- Team roles are not well defined;
- Team-members individual involvement.

Process related risk factors are:

- No information available from the application field (especially for new, innovative applications);
- Poor communication between the team members and between the team and stakeholders;
- Company certification level;
- Suppliers competences;
- Validation and verification process implementation;
- Deliverables quality;
- Change management preparation.

Hardware and software related:

- Software incompatibilities;
- Mobile devices incompatibilities;
- Network bandwidth;
- Software and hardware failures;
- Existing bugs in software;

- Different behavior on emulators and mobile devices.

Environmental related risk factors:

- Users and customers skills and expectations;
- Competitors involvement;
- Stakeholders objectives.

The boundaries between the risk factors classes are not so strictly defined. Some risk factors could influence other factors.

4 Time/Duration Management Models

4.1 Definitions

A project is "a temporary endeavor undertaken to create a unique product, service, or result" [13]. By adapting the definition from [14] we state that an m-application software development project is a temporary endeavour undertaken to create a unique m-application. High quality m-application software development projects deliver the required product within scope, on time and within budget. It is the project manager's duty to skilfully balance the competing demands for project quality, project duration and cost of resources in order to be able to deliver the software as planned.

Like any other type of project, software development projects need:

- clearly defined requirements and scope
- established achievable objectives
- controlled resource allocation
- good effort and schedule management

The expectations of stakeholders are focused on the software to be delivered, on the budged consumption and on the project duration.

The duration of a project is the time elapsed between the project start and the project delivery date, when the software is delivered to the customer. The project duration is an essential indicator that should be well estimated, agreed upon with the stakeholders and thoroughly monitored, up to project completion.

Project duration and size reflect the manager's own understanding of the requirements. It is not possible to correctly size and estimate duration for a project that is not completely understood. Further, project duration provides an important check for scope creep throughout the project. Failing to pay attention to project duration one could agree to add new functionality without appropriately updating project size and effort needed.

4.2 The difficulties of estimating software project duration

There are several reasons that make m-application project duration estimation a difficult problem. First of all, the very essence of software building process makes it difficult to measure. It is a tough endeavour to try to measure "how much" software is there in a software project because the software is invisible and unvisualizable [15]. This especially difficult if we try to make such forecasts before a detailed software design.

The software is pure thought-stuff, infinitely malleable [15]. Unlike cars and buildings, the software is constantly subject to pressures for change because the costs of modifications are difficult to understand.

Many of the classic problems of developing software products derive from this essential complexity and its nonlinear increases with size. From the complexity comes the difficulty of communication among team members, which leads to product flaws, cost overruns and schedule delays. From the complexity comes the difficulty of enumerating, much less understanding, all the possible states of the program, and from that comes the unreliability [16].

4.3 Duration estimation techniques

The grand majority of techniques for m-application project development duration estimation can be found either in bottom-up or top-down category.

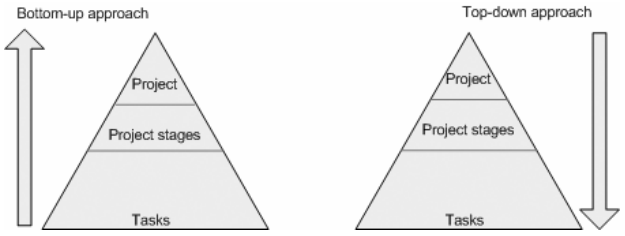

Fig. 3. Bottom-up vs. top-down techniques

The difference between the two comes from the approach used to estimate project duration. The techniques in the first category start at the task-level view of the project and aggregate the work to be performed on higher levels, up to the project as a whole. The top-down way offers

duration predictions based on properties of the work-product, the project team, and the project environment, figure 3.

4.3.1 Bottom-up techniques

This type of duration estimation techniques start with developing a work breakdown structure of the work and then continue with task identification and task duration estimation. Every task should be simple enough so as one could easily answer the question regarding the task duration three parameter estimates:

- best duration estimation
- most likely
- worst duration

Also for every task one should know:

- what is involved in getting started
- how will resources be allocated
- what exactly are the conditions to be met in order the project to be considered done.

The next step is identifying the predecessor-successor relationships and the critical path through the activity graph.

In order to forecast the completion time, three different approaches can be used:

a) The simple approach consists in adding-up the most likely estimates for each task on the critical path. It is not the best method, but it is the simplest one.

b) The second approach means to calculate the expected task duration ED as a weighted mean of the three given estimations using PERT equation:

$$ED = \frac{BD + 4 * MD + WD}{6} \quad (1)$$

where:

BD – best duration estimation; this is the most optimistic expectation, the best case scenario that assumes no influence is going to negatively impact the project duration;

MD – most likely; the duration of activity given the resources, their productivity and realistic expectations of availability;

WD – worst duration; the duration of activity based on a worst case scenario of what is described in most likely estimate.

c) The third approach relies on a Monte Carlo simulation over the task estimation data. The result will be a probabilistic distribution of the project duration [16].

4.3.2 Top-down techniques

Top-down techniques use instead some high level attributes of the project (related to its complexity, functionality or size) and of the organization capability to deliver the project.

Top-down estimation begins with an assessment of the size of the work-product being planned. This idea comes from construction projects, where the project-manager would not imagine committing to a deadline without establishing and tracking some good size estimates, such as the number of square feet, number of windows, doors, etc. to be designed and built.

Up to date there are four software project sizing legacy methods. See table 2 [18]:

Table 2. Project sizing techniques pros and cons

Sizing Method	**Pros**	**Cons**
Lines of Codes	Easy to measure in many development environments (after there is code).	Cannot be done before there are lines of code.
Function Points	Can be measured during requirements stage.	Requires some training, calibration and perhaps tailoring to specific application domains.
Use-Case Counting	Can be measured during requirements stage.	New method. Small experience base at this time.
Web Application Proxies	Easy to count starting with early web application prototypes.	New method. Requires development of counting rules and calibration for specific application types.

The next step in top-down estimation is to use a project duration estimation model.

Lawrence Putnam proposed a widely used model for project duration estimation using data on size, effort, and historic duration for thousands of other

software projects. The model builds up the organization's delivery capability index using PP - *Productivity Parameter* and links it to size, effort and duration dynamics.

$$PP = \frac{PS}{(\frac{E}{\beta})^{1/3} * D^{4/3}} \quad (2)$$

where:

PP – Putnam's productivity index. This item shows the organization's project delivery capability;

PS – project size, counted using one of the above sizing methods;

E – effort (in man-years). The work needed in order to fulfill the project;

D – the project duration (years).

The following things are notable in regard to this model:

a) an organization with higher PP can deliver more size with less effort and in shorter duration than one with a lower PP;

b) the 1/3 and 4/3 exponents in equation 2 express the non-linearity in effort-duration relationship.

4.4 Choosing a project duration estimation technique

Both top-down and bottom-up approaches proved to be good at estimating project duration. A good software project manager will probably use both methods, plus his own estimation, based on priori experience. Bottom-up estimates use work-breakdown structure, critical path method and task estimates; they provide crucial details regarding the duration of smaller project parts and they roll up to a global duration and effort estimation. Top-down estimates rely on history of other real projects. One's cumulative experience in similar projects can provide estimates that deserve some consideration in balance with the bottom-up and top-down views.

5 M-applications Development Performance Metrics

Poor project management is the number one factor leading to failure of IT projects, including m-applications development. Upon completion, a project can meet all the objectives and still be a financially unprofitable project.

High quality project deliverables cannot be obtained without high quality development processes, but a quality process does not guarantee quality products. The quality of the process is certified through quality standards.

Also, well-trained personnel do not guarantee the quality of deliverables. In order to obtain quality results, the organization must have trained and skilled personnel, and standardized project management and technological processes.

A balance must be obtained between: resource allocation for projects, risk and profit, long-term and short-term projects, research and development projects, internal or external projects. Figure 4 describes an organization having four projects with varying degrees of risk, value and profit. In [19] several indicators were proposed for IT project performance measurement. These indicators can be applied to measure the performance of m-applications development projects.

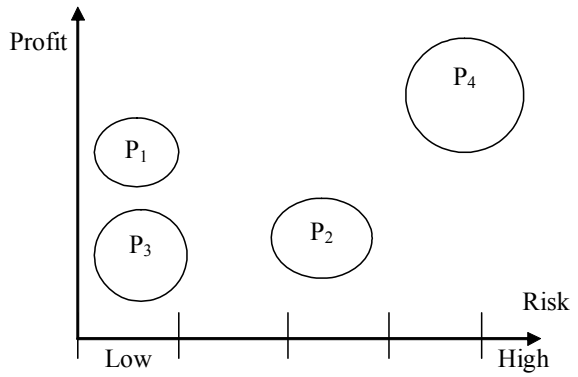

Fig. 4. Projects chart by risk and profit

The degree of objective achievement is calculated as:

$$GA = \frac{OA}{TO} \quad (3)$$

where:

OA – the number of achieved objectives

TO – the total number of established objectives

If the indicator value is greater than one, is considered that the project achieved more objectives than were planed initially.

The ratio between the achieved deliverables and the planned deliverables can be also calculated for each project phase, where deliverables from one phase are inputs for the next phase.

The *degree of satisfaction* can be computed as:

$$DS = \frac{\sum_{i=1}^{p} DSR_i}{TR} \quad (4)$$

where:

DSR – the degree of satisfaction for the requirement i

TR – total number of requirements

p – the number of requirements

The degree of satisfaction for a customer requirements is a value from 0 (no satisfaction) to 1 (fully satisfied) or using a similar scale. The degree of client satisfaction with an m-application can vary with the mobile devices the application is run on.

Work productivity based on inputs is given by:

$$W_1 = \frac{\sum_{i=1}^{n} O_i}{\sum_{j=1}^{m} Ij} \quad (5)$$

where:

Oi – the output *i*; (deliverables, results)

Ij – the input *j* (work, resources per time unit)

n – the number of outputs

m – the number of inputs

Work productivity based on time:

$$W_2 = \frac{\sum_{i=1}^{n} O_i}{T} \quad (6)$$

where:

T – period of time

The *cost of resources* takes into account the category of resources and the cost per unit for each category:

$$C = \sum_{i=1}^{w} NR_i \, d_i \, p_i \quad (7)$$

where:

NR_i – number of resource from the category i

pi – price per unit for the resource category i

di – units of usage for the resource category i

The *total cost of a project* can be defined as:

$$C_T = \sum_{i=1}^{k} c_i \quad (8)$$

where

k – the number of project phases

c_i, - the cost of all resources from the phase i

The number of reworks because of no concordance between the specifications and the results measure the team performance in doing their work.

For the executives, it is important to know the value of all running projects. A *project portfolio value* at a given time is computed as:

$$PPV^s(t) = \sum_{i=1}^{k_s} VP_i^s(t) \quad (9)$$

where:

$PPV^s(t)$ – project portfolio *s* value at the given moment of time *t*

VP_i^s – the value of project *i* from the portfolio *s*

k_s – the number of projects in the portfolio *s*.

Other indicators are developed to measure the performances of IT projects, having in mind the m-applications characteristics. In order to use them, data must be collected from various projects and must be validated.

6 Extended Metrix Model

6.1 Forecasting project duration with Metrix Model

The Metrix model is a hybrid type of model for estimating the duration of software projects [20]. 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 automatically calculates the simulation input parameters starting from easily available data. Also, the model produces an interval for the possible project durations and a probability distribution and not single point estimation. Thus, one is able to know the possible project durations together with the probability that certain duration will materialize.

The components of Metrix model are described in figure 5:

a) an expertise-based component: task duration estimation is performed by the software developer himself who will be responsible with the task completion;
b) a learning oriented component: individual task duration estimations will be automatically adjusted with historical

individual estimation errors, this way enhancing the accuracy of estimations;

c) a mathematical-statistical component: the Monte Carlo simulation is used in order to produce a distribution of probability for the possible project durations;
d) an algorithmic component: the model has input data, it iteratively executes several steps and ramifications and in outputs clearly defined results.

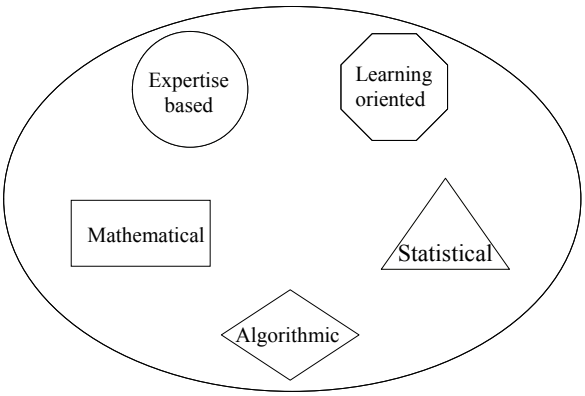

Fig. 5. The components of the Metrix Model for software project duration estimation

From the approach used viewpoint, this is a bottom-up model that takes task duration estimations as input, it aggregates the task into project stages and then it combines them into the project as a whole (see figure 6):

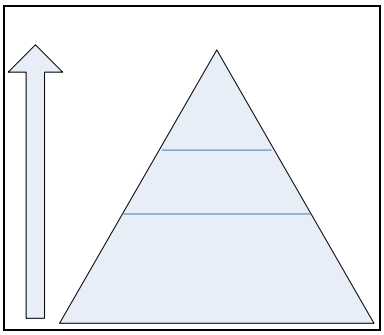

Fig. 6. Hybrid model with bottom-up approach

As follows, the Metrix model structure and the steps it encompasses are presented in greater detail.

Individual task duration estimations and task interdependency represent the input data of the model. The model will also get the history of the duration estimations for the tasks that have already been finished.

The result of running the model is a probabilistic distribution of the project duration. The steps performed are described here under:

Step 1. The historical task duration estimations are collected for every developer. Will be considered both current project finished tasks and the tasks finished in other projects during the last 6 months.

Step 2. For every historical task duration estimation from step 1 we calculate the Estimation Accuracy Index (EAI) using the following formula:

$$EAI = \frac{AD}{ED} \quad (10)$$

where:

ED – estimated task duration (in hours);
AD – actual, elapsed task duration;
EAI – Estimation Accuracy Index.

If EAI is greater than 1, then the task was underestimated, meanwhile if EAI is less than 1, then the task has been overestimated.

Using the results above we calculate the discreet probability distribution for the EAI indexes for every developer part of the team.

Step 3. We build the activity graph using the task dependency and estimated task durations. See figure 7 for an activity graph example.

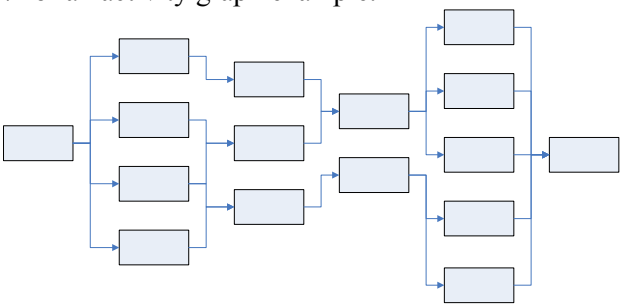

Fig. 7. Activity graph example

Step 4. We find the critical path through the graph and we calculate the deterministic duration of the software project.

Step 5. We run the Monte Carlo simulation. The following operations are performed at each stage:

a) For every task we randomly choose (using the probability distribution from step 2) an estimation error from the same developer's estimation error history. Then we adjust the actual estimated duration with this EAI.
b) We recalculate the critical path method and the project duration.

We repeat the simulation 1000 to 10000 times.

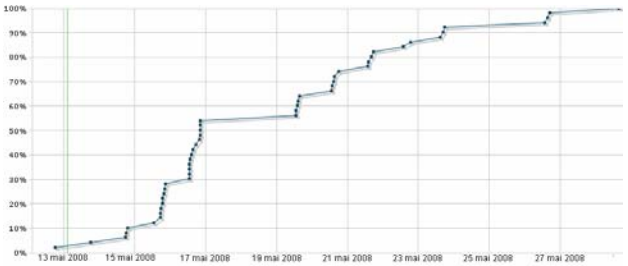

Fig. 8. Probability distribution for project duration and project deadline

Step 6. We calculate the project duration frequencies obtained as a result of Monte Carlo simulation. We display the project duration probability distribution. See figure 8.

6.2 Schedule risk management with Extended Metrix Model

The basic Metrix Model described in the previous section is augmented with risk analysis elements. The resulting model is called the Extended Metrix Model and offers the manager of m-application development projects additional information regarding the schedule risks. More specifically, the new model introduces a new step at which three risk related indexes are calculated for every task: criticality, sensitivity and cruciality.

The criticality index of a task represents the probability that this task will be on the critical path [21], [22]:

$$TC = \frac{\sum_{i=1}^{N} TC_i}{N} \quad (11)$$

where:

TC – task criticality, a number between 0 and 1 inclusively.

TCi – equals 1 if task is on critical path at iteration i and 0 otherwise.

N – the total number of Monte Carlo simulations.

The closer to 1 is TC for a given task, the higher the probability that that task will be on the critical path. The closer to 0 is a task's TC, the higher the probability that the task will not reside on the critical path. The higher the TC of a task, the higher is the importance to manage the duration of that task in order to avoid project delays.

The sensitivity index of a task represents the correlation between task duration and the overall project duration. In practice, the sensitivity index SI is calculated as the Spearman's Rank Correlation between task duration and project duration:

$$SI = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \quad (12)$$

where:

SI – sensitivity index of a task;

$d_i = x_i - y_i$ – the difference between the ranks of the corresponding values x_i (task duration) and y_i (project duration);

n – the number of simulations performed.

The sensitivity index SI values lie between -1 and 1. In the field of project duration estimation, a SI less than 0 has no sense because the project duration cannot be shorter as long as the task duration goes longer. So the only meaningful values are between 0 and 1 inclusively. The greater the SI of a task, the higher is the correlation between task duration and the overall project duration.

The cruciality index CI represents the product of the two indexes calculated above and shows the importance to manage the duration-uncertainty of an activity:

$$CRUI = CI \times SI \quad (13)$$

where:

CRUI – the cruciality index of a task;

CI – criticality index of a task;

SI – sensitivity index of a task.

The CRUI metric has no unit of measure but its significance lies in its ability to rank project tasks according to the descending order of the importance to manage the uncertainty of an activity. The higher the CRUI of a task, the more attention the task needs from the manager of the project regarding timely execution of the task.

In the example below, table 3 lists the top 5 tasks of a project in descending order of their cruciality:

Table 3. Top 5 tasks of a software project in descending order of their cruciality

Task code	Task Cruciality Index
15	0,91
2	0,87
7	0,85
21	0,84
6	0,82

The set of the three calculated indexes, i.e. the Criticality, Sensitivity and Cruciality indexes are

greatly aiding the duration risk management during project management process. The three indexes provide the project manager with critical information regarding the potential individual impact of a task delay upon the entire project duration.

7 Conclusions

M-application software development is an emerging field of the software industry. Despite being a relatively new field, best practices project management techniques can be successfully used to deliver high performance.

The development of mobile applications involves some difficulties engendered by reduced capabilities of mobile devices. Due to mobile devices limitations, in particular limited internal memory and reduced processing power, the source code of mobile applications needs additional optimization which will result in less testability.

M-application project development implies the usage of specific development environments like emulators that are not 100% compatible with the hardware device. This difference requires a slightly different approach both for development and testing.

M-application project duration can be successfully estimated using top-down and bottom-up approaches that have successfully been used over the last decades.

In order for the m-application to be evaluated as successful, a quantitative approach can be employed by the use of a set of performance metrics.

In order to achieve the quality requirements, the mobile applications have to be tested. A comprehensive testing leads to high quality software, but with higher costs [23] and duration overdue.

The presented Extended Metrix Model relies upon the specifics of the m-application development environment. Specifically it uses the widely available historical estimation data to compute task duration probability distribution. The first benefit of the Extended Metrix model is the project risk information associated to every task. The uniqueness of the proposed model is that it determines both the estimated duration of the project and the risks associated with delaying a task. The second benefit of the Extended Metrix model is that unlike classical deterministic models, which offer a single value for the estimated project duration, this model produces a probability distribution of the software project duration. By using this approach we reduce the project uncertainty by allowing the manager to gain better control over the project duration and the associated probability of a certain duration outcome. The third benefit of the Extended Metrix model is that it relies on the historic duration estimation of the team members. Similar models based on Monte Carlo simulations require a duration probability distribution function for every task. This requirement unfortunately set Monte Carlo simulations out of the practical domain into the academic universe. The innovation brought by the Extended Metrix model is the elimination of the probability distribution functions requirement and the use of discreet probability distribution of the EAI (defined in this paper). The EAI probability distribution can be easily determined using the historical estimation errors which are available to most software companies.

Further research will be focusing on the use of prepackaged m-components as a means of speeding up the development process. Also, m-application project development success will be measured by assessing the quality of the m-application user interface.

References:

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

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

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

[4] Ambler S., *Software Development Project Success Survey 2008*, Dr. Dobbs Journal, February 2009

[5] ***, Total mobile subscribers top 1.8 billion, *Informa Telecoms & Media Research Report* http://www.mobiletracker.net/archives/2005/05/18/mobile-subcribers-worldwide

[6] ***, Global cellphone penetration reaches 50 pct, *Reuters*, http://www.reuters.com/article/marketsNews/idINL2917209520071129?rpc=44m

[7] ***, *Key Telecommunications Statistics*, Office of the Telecommunications Authority in Hong Kong

[8] ***, *Europeans hang up on fixed lines*, BBC News,

http://news.bbc.co.uk/2/hi/technology/7116599.stm

[9] Toma C., Popa M. and Boja C., Smart Card based Solution for Non-Repudiation in GSM WAP Applications, *WSEAS TRANSACTIONS on COMPUTERS*, Issue 5, vol. 7, 2008, pp. 453-462

[10] Pocatilu P., *Mobile Applications' Quality Metrics,* Proceedings of International Conference on Business Information Systems, InfoBUSINESS, Alexandru Ioan Cuza University, 26-27 October 2006, pp. 114-121

[11] Diaconita V., Botha I., Bara A., Lungu I.,. Velicanu M., Two Integration Flavors in Public Institutions, *WSEAS TRANSACTIONS on INFORMATION SCIENCE & APPLICATIONS*, Issue 5, Volume 5, 2008, pp. 806-815

[12] Boja C., Bătăgan L., *Software Characteristics of M-Learning Applications* in Proc. of. 10th WSEAS International Conference on Mathematics and Computers in Business and Economics (MCBE'09), Prague, Czech Republic, March 23-25, 2009, ISSN: 1790-5109, ISBN: 978-960-474-063-5, pp. 88-93;

[13] *** - *A Guide to Project Management Body of Knowledge Third Edition*, Project Management Institute, 2003

[14] Vetrici M., Reducing Software Projects Duration using C#, *Informatica Economica Journal*, Vol. VII, No. 1, 2007, pp. 91-95.

[15] Brooks Jr. F. P., Essence and Accidents of Software Engineering, *Computer Magazine*, Vol. 20, No. 4, 1987, pp. 10-19.

[16] McConnel S., *Rapid Development,* Microsoft Press, 1996

[17] Vetrici M., *Project schedule using Monte Carlo simulation with discreet probability distribution*, Proceedings of the 4th International Conference for Applied Statistics, Bucharest, Romania, 2008

[18] Hallowell D. L., *Software Project Management Meets Six Sigma*, http://software.isixsigma.com

[19] Pocatilu P., IT Projects Performance Indicators, *Economy Informatics*, EISSN 1842-8088, vol. VII, No 1-4, 2007, pp. 113-117

[20] Vetrici M., *Software Project Duration Estimation Using Metrix Model*, Informatica Economica Journal Vol. XII, no. 47/2008, pp. 87-91.

[21] Kwak Y.H., Ingall L., *Exploring Monte Carlo Simulation Applications for Project Management*, Risk Management Palgrave Journals, Vol. 9, 2007, pp. 44–57.

[22] T. Williams, *The contribution of mathematical modelling to the practice of project management*, IMA Journal of Management Mathematics Vol. 14(1), 2003, pp. 3-30.

[23] Lazic L., Mastorakis N., Cost Effective Software Test Metrics, *WSEAS TRANSACTIONS on COMPUTERS*, Issue 6, Volume 7, June 2008, pp. 599-619