PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 47, NO. 3–4, PP. 213–228 (2003)

UML BASED SOFTWARE PROCESS MANAGEMENT

Orsolya DOBÁN and András PATARICZA

Department of Measurement and Information Systems
Budapest University of Technology and Economics
H–1521 Budapest, Hungary
e-mail: \protect\LY1\textbraceleftdoban,pataric\protect\LY1\textbraceright@mit.bme.hu

Received: Oct. 1, 2003

Abstract

The main objective of software project management is to assure that a software product will be delivered in time, keeping the cost limits and a proper quality. The key problems are the proper estimation of the effort needed to implement a specific design, the sufficient and effective allocation of resources and the development environment. An appropriate project plan has to have a good or optimal scheduling of the individual development sub-tasks. Finally, a project management methodology has to cope with the risks evolving during the project. These project management activities have to start in the very early phases of the development process in order to keep the deadlines and have to continuously accommodate with the progress of the development process.

UML, the Unified Modelling Language [1] is increasingly widely used to design applications in a very broad range of software products. One of the major benefits of using UML as a design language is that it can be thoroughly used from the very initial phases to the implementation. Exploiting the property of the UML by which it's able to model also the dynamic behaviour of a system, it can be interpreted for workflows, this way can be used to describe also the development process itself.

In this way our main objective was to provide a methodology, which is able to take into account all the main factors influencing the project efforts and scheduling for UML based design, and to generate the mathematical model of a software process optimization problem based on UML diagrams.

Keywords: UML, software development, cost estimation, process optimization.

1. Introduction

A radical growth in software complexity was noticeable in the field of information technology in the former decades. Cost efficiency and development time became the most important factors of the software development process according to the international trends.

While project management methodologies traditionally support the assessment of the feasibility of a software development plan in the terms of time and human resources, only a minority of methodologies support the estimation of the cost factors related to this process. This way cost prediction remains a difficult problem till yet, despite of the use of standard, easy to understand modelling languages.

The current paper:

- gives a short overview on the main principles of the model-based cost estimation models, recapitulating the main features of one of the most widely used cost estimation model (COCOMO II);
- demonstrates the main challenge of integrating the UML based software development environment and the effort prediction process pointing out the main advantages of the automated cost estimation;
- provides an overview about the aspects of the software development process optimization problem, and
- presents an UML based integrated environment adequate to carry out automated cost estimation and development process optimization procedures.

2. Cost Estimation

According to the difficulties sketched above, intensive research is going on to give an adequate cost prediction. A variety of cost estimation models was developed in the last two decades, including commercial and public models as well. All of these major cost estimation methodologies like COCOMO II. [2] or industrial ones like CostExpert [3] assess the estimated project cost from the following main factors:

- the complexity of the software in the terms of its functionality (like function point analysis assessing the number of source lines of code, size of the database, etc.)
- the development process including organizational aspects, life-cycle models, etc.
- the human factors characterizing the skills of the development team.

These cost estimation methodologies derive an own extrapolation formula on the basis of a set of real project time and effort log data.

2.1. COnstructive COst MOdel (COCOMO II.)

Our objective was to select for our pilot experiment a cost estimator having its algorithms open for the wide public. This way COCOMO II., one of the most popular model well accepted in the practice was used.

The COCOMO II. model is the result of the evaluation of a large number of project logs and the estimation of a best fitting empirical curve to their factors.

From this point on, we refer by COCOMO both the original method [4] and its refinement known as COCOMO II.

The main formula of COCOMO expresses the predicted effort in units of the *Person Months (PM)*, amount of time one person spends working on the software development project for one month *Eq.* (1).

$$PM = A \cdot \left(\prod_{i=1}^{17} EM_i \right) \cdot \text{Size}^B \tag{1}$$

Its inputs can be divided into three categories (*Fig. 1*):

- 5 *scale-drivers*, which are specific to the development process, and determine the value of the exponent *B* in the main COCOMO II formula;
- 17 *effort-multipliers* (EM), related to the target software product and to the development environment;
- the estimated *Size* of the software to be developed in units of thousands of source lines of code (KSLOC). The goal is to measure the amount of the intellectual work put into the program development, but difficulties arise when trying to define consistent measures for different programming languages. Fortunately, additionally to the direct, heuristic estimation of the code length the function point (FP) based prediction can be used as well. FP extrapolates the code size from the number and complexity of the product's designated functionality from the system requirement list.

The multiplier *A* is an empirical constant serving the best fit of the curve to the logs.

Size	Scale drivers	Effort multipliers
	Precedentness (PREC)	Product factors
	Development Flexibility (FLEX)	Software Reliability (RELY)
	Architecture/Risk Resolution (RESL)	Database Size (SIZE)
	Team Cohesion (TEAM)	Product Complexity (CPLX)
	Process Maturity (PMAT)	Documentation (DOCU)
		Required Reusability (RUSE)
		Platform factors
		Platform Volatility (PVOL)
		Execution Time Constraint (TIME)
		Main Storage Constraint (STOR)
		Personnel factors
		Personnel Continuity (PCON)
		Applications Experience (AEXP)
		Analyst Capability (ACAP)
		Programmer Capability (PCAP)
		Platform Experience (PEXP)
		Language and Tool Exp. (LTEX)
		Project factors
		Use of Software Tools (TOOL)
		Multi-site Development (SITE)
		Development Schedule (SCED)

Fig. 1. Input parameters in the COCOMO II model

The large number of multipliers takes advantage of the greater knowledge available in the later development phases, to support gradually refined estimations.

Each factor has an associated range of rating levels (‘very low’, ‘low’, ‘nominal’, ‘high’, ‘very high’, ‘extra high’). COCOMO II assigns to each qualitative category a corresponding empirical numerical value. The first step of the cost estimation in a new project is the classification of the factors into one of these categories.

The first factors related to the application design can be extracted from the UML product design itself. Here the basic idea is to assign costs to the different fragments of the target product which can be done either by an expert estimator in a heuristic way or in an automated way derived from the complexity of the UML sub-diagrams and synthesize a system wide model from these [5].

In the subsequent sessions we deal with the estimation of the five scale drivers and 17 effort multipliers which depend on the development process, development environment and on the product itself.

3. UML Integrated Cost Estimation

Since UML is increasingly widely used for object-oriented software product design as a standardized formal modelling language, our aim was to exploit this property in the field of project management. Obviously, a common graphic language accepted by every member of the project team can be extremely useful for mutual understanding, moreover if this language is identical with that used for product development then a single environment can capture both the product and the development process features.

The traditional way of cost estimation is to derive its input factors from three different sources (*Fig. 2*).

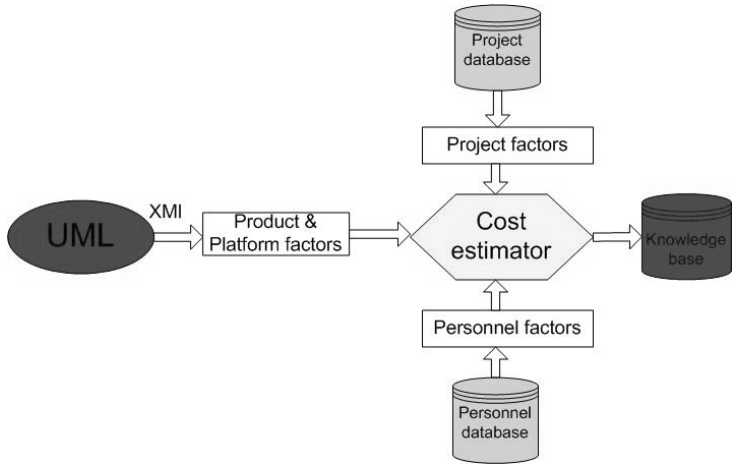

Fig. 2. General way of cost estimation

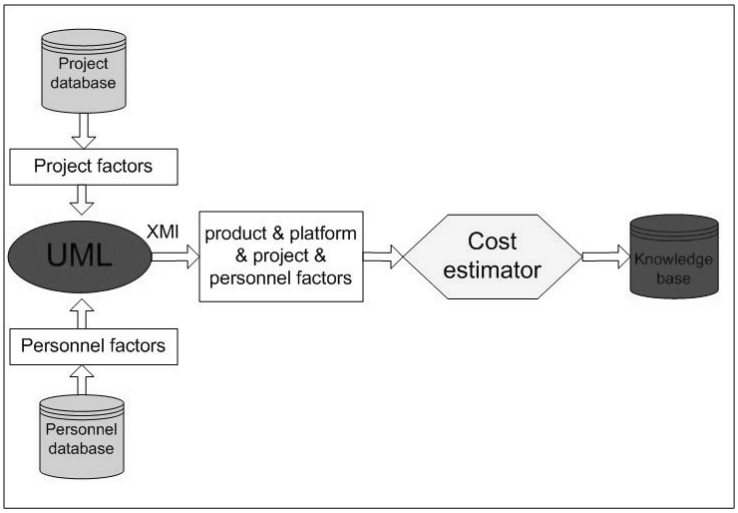

Fig. 3. Integrated way of cost estimation

One possibility is to determine the project factors on the base of a project database containing empirical data of former processes, and to derive the personnel data from a personnel database. Such a personnel database has to contain skills and experience characteristics of the participating members in the development process. The product and the platform related factors can be derived directly from the UML based models. The result of the cost estimation is usually stored in a knowledge base. This database can serve as a starting point for ongoing cost estimations. Such an empirical database can contribute to realistic cost predictions.

Our aim was the use of UML to describe the target product and the development process itself. The use of an identical modelling paradigm supports the integration of all important factors into a single model (*Fig. 3*). From this model a cost prediction method can be used to automatically derive effort estimations. During the progress of the development process the product specification is gradually refined, the efforts made to finish the project cardinally decrease this way a gradually refined series of cost estimations can be produced.

The goal of the integration was

- to decrease the additive effort needed to determine the cost related factors during the UML based software development;
- to support change management if product has to be altered, by delivering cost estimator for the modification process.

To realize this integration the main tasks were

1. to enrich the UML model with the input parameters needed for the cost estimation. This had to be done in a well defined, unique way;

2. to extract the needed, cost related information from this extended model.

In this integrated environment the standard UML modelling language is used to describe the target software product. As UML profiles were created to use the UML language in different fields of application areas, an OMG standard UML profile is used to model the software development process.

OMG did elaborate a standard profile to describe the software process engineering metamodel (SPEM) [6]. SPEM is adequate to describe the workflow of the development process in UML (*Fig. 4*).

One of the main insufficiencies in the current SPEM is that only little support is offered for the quantitative characterization of the software development process. For this reason we did integrate the standard SPEM with the General Resource Model (GRM), originally introduced for the quantitative characterization of the resource usage in real-time systems [7].

GRM describes the application-resource interactions in the terms of services required by the application and offered by the resources of the underlying platform. Obviously, in a software development process the resources are the developers, who offer their services to the software development process. The service required from the developer is to perform some activities during the process for instance to develop some codes in a given language, or to test a module.

The quantitative characteristic assigned to this development process is the time needed to perform the task. The GRM profile is used to model both the human resources of the software development process and the infrastructure available during the project. The resource model is parametrized by the human factor values which are usually stored in different personnel databases at enterprises.

Fig. 5 describes the main concept of our project management framework. The product model is modelled in standard UML, where the package diagram describes the product modules, the individual parts of the software which can be developed separately from the other ones. Several factors related to the product need information on the platform like platform volatility, execution time or main storage constraints. In this way, additionally to the product model, these factors are modelled by the standard UML GRM.

The effort-related parameters assigned to a software package are:

- product factors (RELY, SIZE, CPLX, DOCU, RUSE)
- four of the scale factors (PREC, FLEX, RESL, TEAM)
- estimated source lines of code (SLOC)
- the module's programming language (LANG)
- applied developer tool (TOOL)

Two additional parameters have to be determined at the software module level, as the

- type of application (APPL) which is an important reference point at the qualification of the developers' experience;
- module's priority (PRIO), determining an importance factor, a priority for the sub-product. This factor is only used during the process optimization, and it is out of the scope of the COCOMO model.

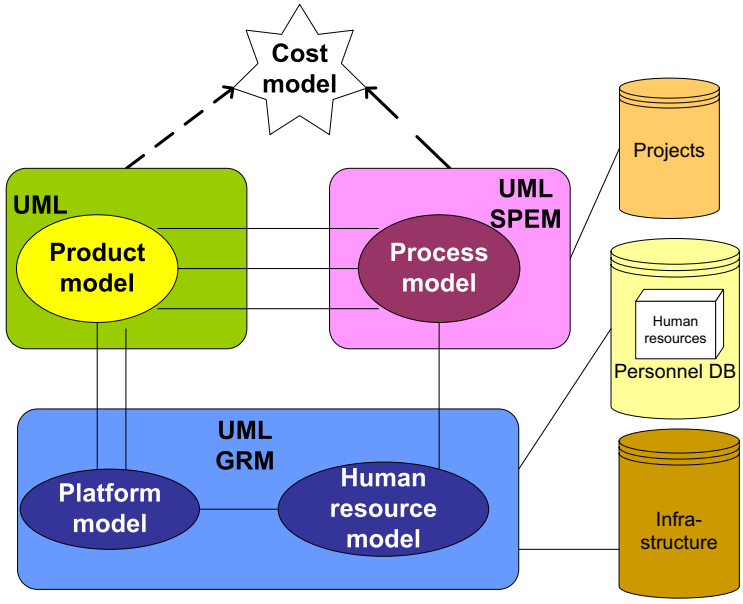

Fig. 4. Model integration in UML environment

The SPEM UML profile describes the development phase of a given software module by the means of an UML activity diagram. This development phase is modelled according to one of the most widely used life-cycle models, the Controlled Iteration Model which determines four sub-phases of the software engineering process, like inception, elaboration, implementation and transition.

To every unique software process sub-phase a developer is assigned whose model element defines the required personnel conditions and properties (estimator's personnel factors, except the personnel continuity [PCON] factor characteristic for the whole group of the developers) needed to fulfil the corresponding tasks. In this way the project manager can determine the experiment and skill level which is required for the developers or for a group of developers being responsible for the given subphase.

Two resource models are used, the first one describes the underlying platform for the target applications, the other one models the available human resources together with their skills offered for the development process. The resource diagram connecting to the SPEM workflow model determines the set of developers available during the entire process.

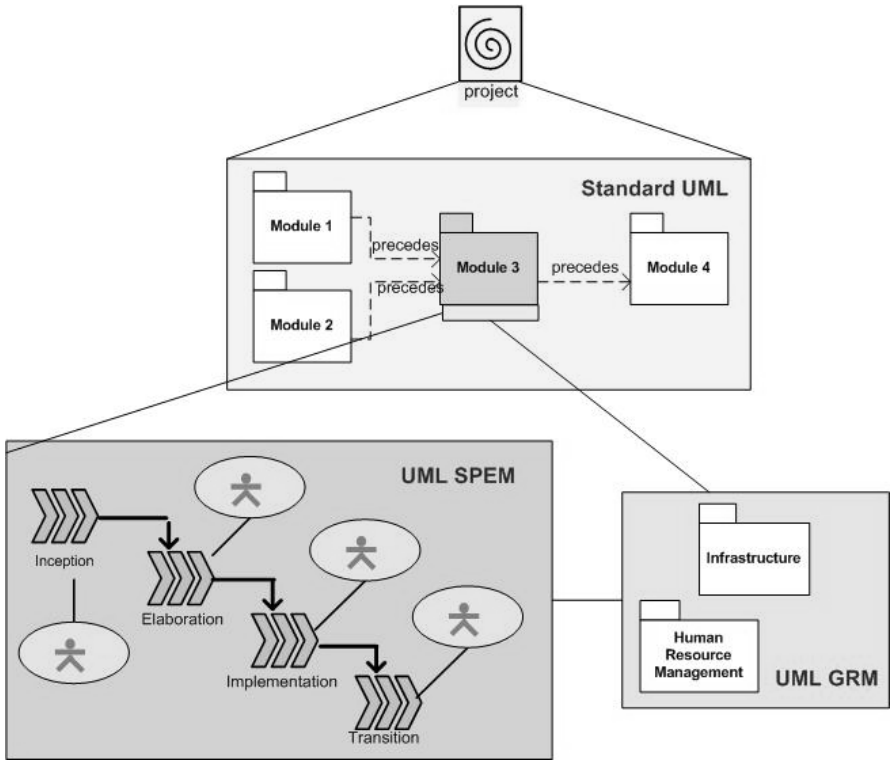

Fig. 5. Model hierarchy

3.1. Technical Realization of the Integration

Fig. 6 summarizes the cost estimation input factors assigned to the different UML diagrams.

As it is shown in the figure the standard UML product model contains the product related parameters, as well as the four scale factors characteristic for every single software module. The number of source lines of code is one of the main input parameters of the estimation model, which can be determined either on a heuristic way by expert prediction or with size estimation models, like function point analysis, which extrapolates the size of the code from the designated functionality of the product.

The additional parameters as the applied programming language, the application category, etc. determine the application field on which the developers' skills are to be classified. The underlying GRM describes the available human resources to which diagram of the personnel factors are assigned. The run time platform is also modelled with the cost related platform factors by the GRM profile.

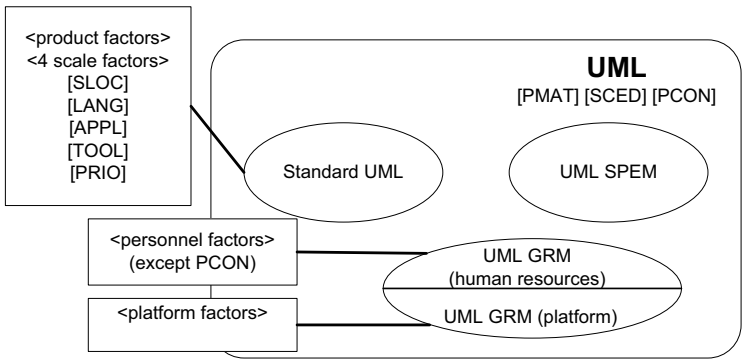

Fig. 6. UML integrated cost factors

The three remaining input factors are:

Process Maturity (PMAT) which factor is organized around the Software Engineering Institute's Capability Maturity Model (CMM). This factor is characteristic for the company carrying out the development process, and in this way it can be taken as a constant for a usual project.

Required Development Schedule (SCED) which rating measures the schedule constraint imposed on the project team developing the software. The ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. Accelerated schedules tend to produce more effort in the later phases of the development process because important issues are postponed due to the lack of time. Please note, that in our case the scheduling tends to decrease by the automated optimization of the subtasks. In this way this factor has to be estimated in an iterative way starting from an initial schedule set up by the project manager, to estimate the required development schedule factor, by deriving from the time effort estimation, and after a subsequent optimization of the process schedule by the re-evaluation of the corresponding factors.

Personnel Continuity (PCON) describes annual turnover at the company involved in the project. In the case of a high personnel turnover an additional cost can arise, as a series of logically ordered activities will be carried out by different persons.

Fig. 7 shows the way of the UML based automated cost estimation.

The source of the cost estimation is the personnel and project database which serves as a knowledge base containing the former empirical, and the human resource related data. After deriving the necessary empirical data from this database all the needed parameters of the cost estimation are included in the different UML diagrams in identical environment. As the standard UML metamodel elements can be extended by UML tags, these parameters are assigned to the diagram elements in

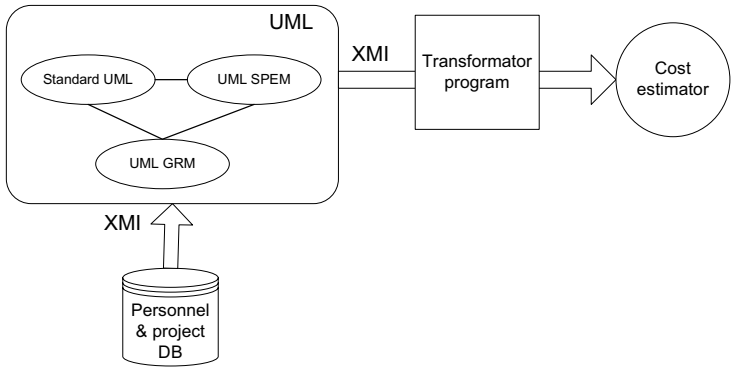

Fig. 7. Automated cost estimation

the same way. The hierarchy describing the entire development process (including also the product package diagram) is stored as a template, in this way it can be imported to any UML based environment. Consequently only the product diagrams have to be created by each individual software development project, the process and resource diagrams can be reused inside a given company.

The transformer program has the function to take the standard UML model in the standard XMI format, to filter out from it the information relevant to cost estimation and to perform a syntax transformation to the input language of the cost estimator.

3.2. The Advantages of the Integration

As described above, extended UML by SPEM and GRM is able to completely describe the development process both in quantitative and qualitative terms, this way

- the automated cost estimation can reuse the existing methodology at the enterprise (please note that COCOMO is one of the richest cost estimator models in the terms of input factors taken into account);
- if a proper modelling style is used by the developers, some main parameters needed to cost estimation can automatically be derived from the UML diagrams created during the functional development process. One example is the derivation of the complexity parameter from class diagrams, or the derivation of the SLOC estimators from use-case diagrams describing the target functionality of the software. This latter example clearly indicates the importance of the proper modelling style as the detail level of use-case diagrams is not uniquely defined in the UML standard.
- The enterprise specific cost constants can simply be substituted into the estimator.

At the same time the uniform use of UML and the possibility to automatically estimate the predicted cost offer additional potential to such a method.

Frequently the designer has to cope with several design alternatives to implement specific functions of the target system. For instance one alternative is to create a generic solution for a part of the target application or to go for component integration and to develop only the necessary interfaces. The fully automatic derivation or the reduction of cost estimation to the expert estimation of smaller modules offers a possibility to compare these design alternatives from the point of view of cost and to automatically select the best approach from the point of view of cost efficiency.

4. Software Development Process Optimization

Another possibility to optimize a development process is to generate a standard mathematical resource allocation and scheduling problem out of it and to solve it by some existing tool. This can deliver an optimal task assignment to the project and the best effort scheduling as well.

The two most important candidate objective functions in the optimization of software projects are the duration of the development and the cost of the development process.

1. As a scheduling problem the objective function is to find the development process of the minimal duration by keeping a predefined budget.
2. As cost optimization problem there is to find the most effective assignment of resources of the lowest cost while keeping the deadline.

For setting up the correct mathematical model of these optimization problems, the basic elements of the scheduling and resource allocation should be identified, namely the software development process has to be described in the terms of optimization units, as

resources correspond to the developers or developer teams participating in the project

activities which have to be sequenced and fulfilled are the four different subphases of every given module development

dependencies among that tasks are determined by the precedence rules of the product modules described in the software package diagram

duration of an activity is calculated by the applied cost estimator for every possible task-resource combination

cost of an activity is the result of the production of the activity's time duration and the corresponding developer's salary rate.

To determine the software development project our system is dealing with, the following boundary conditions were defined.

1. Exactly one developer is assigned to each task, and we do not assume that an elementary activity will be carried out by different persons.

2. The activities are non-breakable, they can not be interrupted by any other task.
3. Every resource (in our case the developers) has a capacity limit which defines the percentage of availability the developer can work on the project.
4. Each activity has a predefined set of candidate developers who are able to perform this task, and the optimization has to select the best developer out of this set.
5. Every product module has a priority level which determines its importance during the development. There are three priority categories
   (a) core module which is essential for the target product
   (b) complementary module which can be sequenced with a less priority during the project
   (c) optional module which can be omitted due to the lack of time or a low budget limit.

Please note that different software development strategies, like extreme programming can be modelled by these priorities.

From these definitions a mathematical optimization problem can be constructed in an automated way from the UML diagrams delivering the cost or time optimal solution.

In our work OPLStudio [12] was applied whose tool provides a high level constraint programming language for the declaration of a mathematical model. Similarly, OPLStudio supports the separate definition of the problem structure in the form of a high level programming language, and the parametrization of this model with numerical values for instance from a database.

According to the first results, in the case of a small scale problem the optimization takes approximately 15,84 seconds with a commercial optimizer engine for a system consisting of 4 modules and 4 programmers. However, dedicated solvers may improve the speed by more than 1 order of magnitude.

Theoretically it is still possible to perform an enterprise level of global optimization which may solve the same resource allocation and scheduling problem by covering multiple, parallel projects as well. However, one of the limiting factors is the computational time, current experiments aim to solve this problem with dedicated solution algorithms.

4.1. Decomposition of COCOMO Model into Factors

The predicted effort has to be calculated for every possible activity-developer combination during the resource allocation problem in order to determine the duration of the different tasks. In the case of complex development processes the number of these calculation steps can be extremely high which could decrease the effectiveness of the optimization process. Our aim was to decompose the different cost factors for the optimization in order to have a more efficient search of the optimal solution.

As already described above, the main equation of the COCOMO cost estimator has a product of terms form. In order to determine the module cost independently from the developer working on it, the main task was to separate the module- and the developer related components of the estimated cost. Taking the logarithm of both sides of the equation, the sum of the estimation's input parameters appear on the right side (*Eq.* (2))

$$PM = A \cdot \left(\prod_{i=1}^{17} EM_i \right) \cdot Size^B$$

$$\log(PM) = \log(A) + \sum_{i=1}^{17} \log(EM_i) + B \cdot \log(\text{Size}) \tag{2}$$

These input parameters can be divided into two groups, the product related components (5 product, 3 platform, 2 project factors) and the personnel components (5 personnel factors). It's important to note that 3 input parameter values (SCED, PCON, PMAT) are either fixed or determined on an iterative way during the optimization process, this way they are missing from the used estimation formula.

$$\log(PM) = \left[\log(A) + \sum_{i=1}^{10} \log(EM_i) + B \cdot \log(\text{Size}) \right] + \left[\sum_{i=11}^{16} \log(EM_i) \right]$$

$$\log(PM) = \log(PM_{\text{module}}) + \log(PM_{\text{personnel}})$$

This way the logarithm of every module cost can be calculated, the individual aspects can be analyzed separately, and the mathematical problem can be derived from the separate part of our UML based model. By the help of these partial results the number of calculation steps needed for the data initialization can be decreased, as instead of 15 input parameters (10 module related and 5 personnel) only the 5 personnel factors have to be scaled and calculated together with the already known module costs.

4.2. UML Based Integrated Environment

The integrated environment of our pilot system (Fig. 8) implements the automated, UML based cost estimation and software development process optimization. As already described above, the target software product is described by a package diagram, the platform conditions are included in the underlying GRM model. The use-case diagram based on the SPEM notation defines the needed skill levels and properties of a developer assigned to a given activity, and the connected GRM diagram contains the available human resources during the project.

The first step of the automated process optimization is to fill the optimization database with the basic elements, as the resources, the tasks and their dependencies.

This information is extracted from the UML diagrams by using the standard interface of the development environment, the XMI description language.

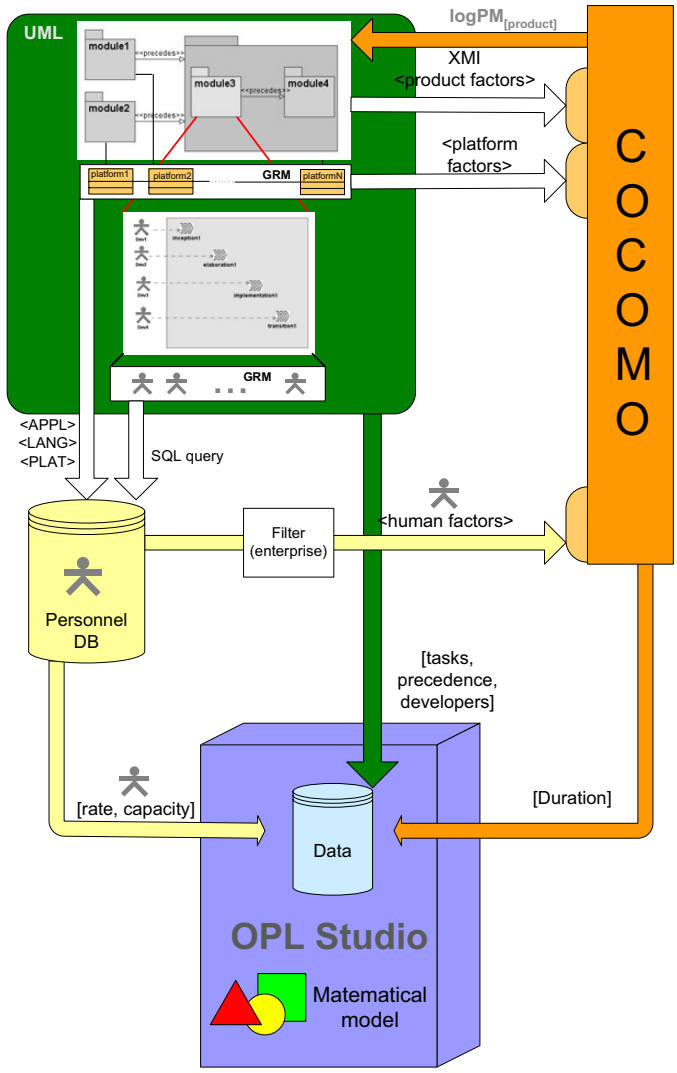

Fig. 8. UML based integrated environment

The next step is to execute a cost calculation for every given module independently from the assigned developer. To estimate the cost of a module the needed product and platform parameters are exported from the UML environment in the standard XMI form, and transformed into the input language of the applied cost

calculator. Using the decomposition method for the cost estimator equation the logarithm of the module cost can be imported back into the UML diagrams.

The following step is a database search for the set of the candidate developers and for their associated parameters. To select the needed parameters upon the personnel database, the type of the software application and platform as well as the applied programming language have to be known, this way they are extracted from the product related UML diagrams. The required properties of the developers assigned to a given task are derived from the GRM human resource diagram. Selecting the related human parameters from the personnel database the estimated effort can be calculated in this case extended by the personnel parameters.

The result of these two estimations is a data matrix containing the time duration of every possible developer-task combination. By selecting also the rating and capacity data of the developers from the personnel database, the cost factor of a task can be computed, and stored in the data initialization database.

After data initialization optimization can be performed. As OPLStudio offers a variety of different visual representations the received optimal software development scheduling can be represented for instance by means of the most popular Gantt diagrams.

5. Conclusion and Further Works

UML is able to model both the target application and the process developing it. The uniformity of the notation allows capturing all the important aspects relevant to project management.

The information fusion and mathematical model transformation technologies allow for an exact optimization of the costs related to the development process.

It has to be pointed out, that this support to the project management can be provided from the very early phases of the design through a series of more and more refined cost estimation as the system design makes its progress and produces more and more fine granular models.

During the software development process optimization an emerging question is the accuracy of effort estimations. The COCOMO model based cost estimations are one of the most widely used estimation methodologies in the USA, and a research is going on in Hungary [9] to test this method in our particular environment. At the same time it is important to emphasize that additionally to the absolute values of the estimators the relative relation between alternate solutions is important as well, as this helps the project management to select the most effective implementation strategy.

In the future we would like to extend our work to handle multiple parallel projects as it is typical at enterprise level, and to exploit the experiments of our actual method which is currently under industrial testing.

References

[1] Object Management Group (OMG), Unified Modelling Language, http://www.omg.org.
[2] BOEHM, B. W., Software Cost Estimation with COCOMO II., Prentice Hall PTR, 2000, New Jersey.
[3] Cost Expert, http://www.costexpert.com.
[4] BOEHM, B., Software Engineering Economics, 1981.
[5] UEMURA, T. – KUSUMOTO, S. – INUOE, K., Function Point Measurement Tool for UML Design Specification, Proc. of the METRICS'99, Boca Raton, Florida, November 1999, pp. 62–69.
[6] OMG Group, Software Process Engineering Metamodel (SPEM), http://www.omg.com.
[7] OMG Group, General Resource Model (GRM), http://www.omg.com.
[8] DOBÁN, O. – PATARICZA, A. – PINTÉR, G., Data Collection through Knowledge Base, IKTA 00194/2000 – 1., Foundation for the Hungarian Higher Education and Research project.
[9] DOBÁN, O., COCOMO Based Cost Estimation, IKTA 00194/2000 – 2., Foundation for the Hungarian Higher Education and Research Project.
[10] DOBÁN, O. – PATARICZA, A. – PINTÉR, G. – SZÉCSI, G., UML Based Cost Estimation, IKTA 00194/2000 – 3., Foundation for the Hungarian Higher Education and Research project.
[11] DOBÁN, O. – PATARICZA, A., Quality Guaranteed System Design, IKTA 00194/2000 – 4., Foundation for the Hungarian Higher Education and Research Project.
[12] OPLStudio: http://www.ilog.com.
[13] OPLStudio User Manual, http://www.ilog.com.
[14] DOBÁN, O. – PATARICZA, A.: Cost Estimation Driven Software Development Process, *Proceedings of the 27th EUROMICRO Conference*, ISBN 0-7695-1236-4, Warsaw, Poland, 4–6 September 2001, pp. 208.
[15] PATARICZA, A. – CSERTÁN, GY. – DOBÁN, O. – GÁBOR, A. – SZIRAY, J., Process Modelling and Optimization in UML, *IEEE International Conference on Intelligent Engineering Systems, INES-2001, Proceedings*, Helsinki, September 16–18, 2001. pp. 457–461.