Software Performance Measures to Assist Decision Makers within the Rational Unified Process

Antonia Bertolino[1], Gaetano Lombardi[2], Eda Marchetti[1], Raffaela Mirandola[3]

[1] Istituto di Elaborazione della Informazione, CNR, Area della Ricerca di Pisa
56100 Pisa, Italy
{bertolino, e.marchetti}@iei.pi.cnr.it
[2] Ericsson Lab Italy SpA, Via Anagnina, 203
00040 Roma, Italy
gaetano.lombardi@eri.ericsson.se
[3] Dip. Informatica, S&P, Università di Roma TorVergata
00133 Roma, Italy
mirandola@info.uniroma2.it

Abstract. Rigorous and automatable approaches to software process measurement are a technology essential for supporting decision makers, such as project managers, in obtaining reliable estimations of relevant properties of the development project and in controlling it. We introduce a methodology, called Propean (Project Performance Analysis), which applies classical techniques of performance analysis to a UML model of the development process. For this purpose, the Real-Time UML profile is considered. The analysis of the UML diagrams produces as an output, in automatable way, the time to completion and the utilization rate of employed resources (personnel in this cases). This paper focuses in the application of Propean to the widespread Rational Unified Process (RUP) adapted to a specific industrial project. We describe the steps necessary for Propean application, the typology of results that can be obtained and how project managers can use them for producing estimations about project completion within fixed schedules and budgets.

1 Introduction

Nowadays software measurement has become a crucial technology in many fields of application and notably in the management of the software development process. We present in this paper an industrial investigation in the usage of software performance measurement to aid managers in making sound, reliable predictions in software development projects and in optimizing the decisions about the utilization of resources, typically the people.

The basis of our study is the metaphor that project teams can be assimilated to the processing elements of classical performance models, and the development activities to the tasks to be performed by the processing elements within established time intervals.

Following this metaphor, we have developed a methodology that takes in input a UML model of the processes to be assessed and through some internal processing steps (by which the UML model is translated into a Queueing Network model that can then be analyzed by traditional performance tools) returns the desired estimates in terms of the time to completion of specified tasks, the utilization rate of involved personnel, and similar indices.

So far we have investigated this methodology for specific project stages, for which we have developed ad hoc UML models [1, 2]. In this paper we focus on the Rational Unified Process (RUP) [11], which is one of the emerging processes adopted in the industrial context. Providing in the little space available a compendium of this process is hopeless and we give up with any attempt to provide a background description of RUP. We assume in fact that the reader has a basic knowledge of RUP and refer to the existing literature for an introduction (e.g., to [11, 17]).

With all its accompanying stuff of documentation, guidelines, supporting tools, templates, etc., RUP in fact constitutes nowadays the most detailed process model and proposes itself as the natural candidate for any organization developing large software systems. But what makes RUP different to the existing process models is its development and packaging in the form of a software product. In contrast with the traditional documentation into big, time consuming and rapidly obsolescent paper manuals, the RUP "product" is available online with extensive use of hyperlinking and cross-referencing, and is regularly updated exactly as software products are.

The intent of this research is to augment RUP with the capability to produce reliable schedule and resource utilization estimates of use to RUP decision makers. If the goal of RUP is to produce high-quality software within *predictable schedule and budget*, we need means to reliably draw such predictions: for instance, how long will RUP take to process a certain project? How will RUP utilize the available resources? How is RUP schedule affected by the concurrent processing of several projects? This paper's contribution is a methodology to answer such questions on rigorous grounds. RUP is therefore equated to a product, as considered in [11], of which we analyze the performance just as we do with any other critical product.

The proposed RUP performance analysis methodology is called Propean (for Project Performance Analysis). To implement Propean, we need to derive a complete UML model of RUP under the recently adopted Real-Time UML Profile, which allows us to annotate the diagrams with time and utilization specific indices useful for our estimation purposes.

In this paper, we outline the Propean methodology (Section 2), and develop a small case study in which we have configured RUP to fit the needs of an adopting organization (Section 3). We illustrate the kind of results that we can obtain in Section 4 and briefly survey related work in Section 5. Conclusions and hints to future work are in Section 6.

2 Description of the Methodology

The Propean methodology consists of two phases: in the first phase we derive the RT-UML diagrams that model the RUP process as configured to fit the needs of the specific organization under exam. This phase pertains to the manager, who has to model the process and the workflows to be instantiated. This may seem a heavy requirement; in practice, the UML process model can be derived once for an organization, and then at each new application of the technique to a specific project, the manager only needs to update the parameters in the diagrams (such as the number of people involved, and the estimated duration of the single activities).

The second phase consists of the processing of the derived diagrams to obtain the performance estimations. This second phase can be automated, and we are currently working to a tool implementation that automatically translates the RT-UML diagrams into a format that can then be processed by classical performance analysis tools (the results presented here are partially obtained by hand translation).

Propean builds on some earlier results that are briefly summarized in Section 2.1; then, in Section 2.2, we outline more in detail the methodology steps.

2.1 Background

We apply well-known techniques from the field of Software Performance Engineering (SPE) [19, 20]. The SPE basic concept is the separation of the software model (SM) from its execution environment model (i.e., hardware platform model or machinery model, MM).

The SM captures the essential aspects of software behavior and is represented by means of Execution Graphs (EGs). An EG is a graph whose nodes represent software workload components and whose edges represent transfer of control. Each node is weighted by use of a demand vector that represents the resource usage of the node (i.e., the demand for each resource).

The MM models the hardware platform and is based on the Extended Queueing Network Model (EQNM) [13]. To specify an EQNM, we need to define: the components (i.e., service centers), the topology (i.e., the connections among centers) and some relevant parameters (such as job classes, job routing among centers, scheduling discipline at service centers, service demand at service centers). Component and topology specification is performed according to the system description, while parameters specification is obtained from information derived by EGs and from knowledge of resource capabilities. Once the EQNM is completely specified, it can be analyzed by use of classi-

cal solution techniques to obtain the usual performance indices such as the mean network response time or the utilization rate.

In our metaphor the project teams correspond to the processing resources in performance models, and the project activities to the tasks to be performed within established time intervals. In other words, using the SPE concepts, SM captures the aspects relative to the activity planning, while MM the ones relative to people (over/under) utilization and distribution.

We applied initially a method proposed in [4, 16] for the derivation of performance models based on SPE techniques, starting from a set of UML diagrams. In this way, the managers using the methodology need not to be expert of performance modeling notations, but they can use the familiar UML language.

The method, in its original conception, used as an input: the Use Case Diagram, to derive the user profile and the software scenarios; the Sequence Diagram, to derive the software model, and the Deployment Diagram, to derive an hardware platform model and to identify the hardware/software relationships. The method then automatically extracted from these diagrams the main factors affecting system performance and combined them to generate a performance model. To be able to model how events and tasks succeed in time, the input UML diagrams had to be annotated with some simple ad hoc labels.

Historically, in fact, the standard UML notation lacked a quantifiable notion of time and resources allowing for the expression of non-functional requirement. By general consensus this UML lack was felt as "an impediment to its broader use in the real-time and embedded domain". The RT-UML Profile has been recently proposed as a response to these needs; RT-UML settles a set of domain profiles for UML allowing for the construction of models that can be used to make quantitative predictions regarding the characteristics of timeliness, schedulability, and performance [18].

Therefore, the natural evolution of our methodology was to customize the input modeling to the standard RT-UML profile, which we describe in [2]. This new version of the methodology is the one that we adapt here to the needs of modeling RUP, and how we do this is described below.

2.2 Propean Applied to RUP

The Propean application to RUP is divided into two steps. The first, called RUP modeling (Sec. 2.2.1), consists of the description of the functionality and architecture of the RUP product by means of UML diagrams appropriately annotated according to the RT-UML profile. The second step, called RUP customization (Sec. 2.2.2), represents the core of the Propean application. The UML diagrams developed in the previous step are refined and completed by the manager, accordingly to personnel availability and process exigencies. In the next subsections the two steps will be briefly described in their main aspects.

2.2.1 RUP Modeling

In this part we present a quick description of the procedure we used to represent RUP applying Propean. As for the development of any other software product, the first step is to describe the system functionalities (i.e. the process activities) in terms of Use Cases (UCs). This description follows an iterative process, incrementing at each iteration the level of detail of the system functional specification. We start therefore representing the interaction of the external actors (in our case, End User, Customer and Stakeholder) first with the different RUP phases (Inception, Elaboration, Construction and Transition), and then with the single workflows. The description is further refined representing the interaction of the external and internal actors (one actor specification per role) with the workflows activities in the different phases. Finally for each activity an annotated Sequence Diagram (SD) representing the roles interaction is developed. The Figure 1 reports an example of the SD used.

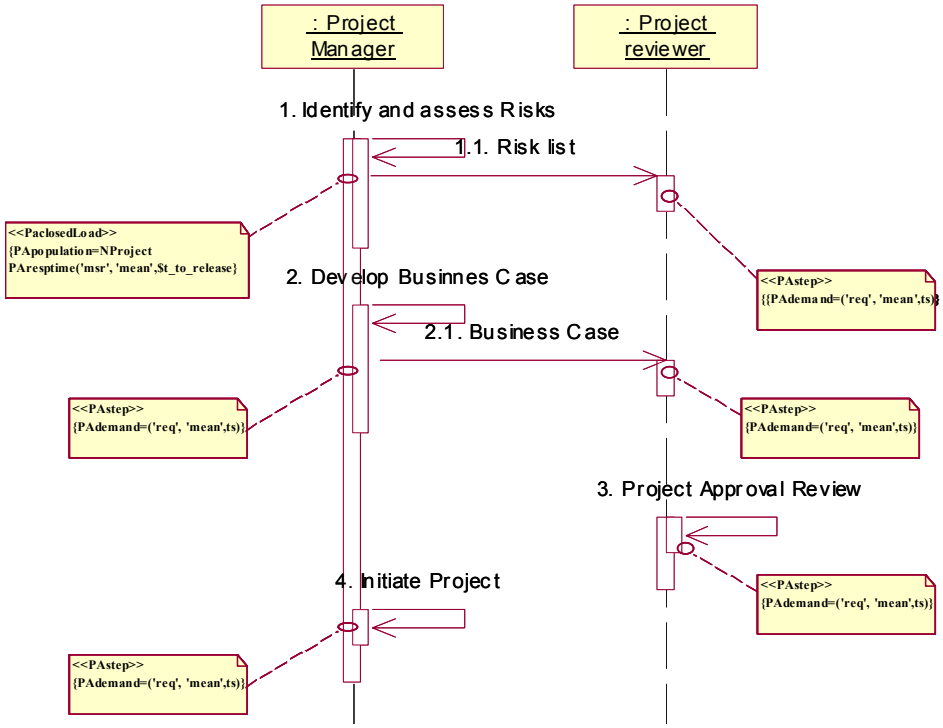

Fig. 1. Activity "Conceive a new project" of the workflow Project Management.

The Propean modeling of RUP proceeds with the identification of the organizational structure of the system, i.e., the architecture definition. As for the system functionality, the architecture definition follows an iterative process describing the decomposition of the system in parts that interact through interfaces, relationships, and constraints. First of all therefore we describe as subsystems the Roles Set called Analysts, Mangers, Developers, Testers and Additional Roles and we define the interfaces they uses. In this case, the attributes of the interfaces are the set of exchanged documents.

The subsystem definition is then refined associating a class to every role and describing the interfaces they use. For every class the attributes represent the artifacts, and the methods are the activities in which the role is involved.

The UML description of RUP derived so far represents only the static structure of the process and is the common starting point for applying Propean to different real situations. The Propean user, typically a manager, starting from this process framework, must adjust and characterize it with respect to the specific needs, peculiarities and constraints of his/her organization. In particular, as described in the next section, he/she has to identify the dynamic structure of the process and express the sequential flow of activities in the different RUP phases.

2.2.2 RUP Customization

In the previous section we briefly explained the incremental process adopted for deriving the UML model of the static structure of RUP. Here we explain the steps to be performed (and specify whether they are up to the manager or are automatable) in order to customize the RUP process to the specific organization existences, and to derive successively a queueing network based model for making predictions.

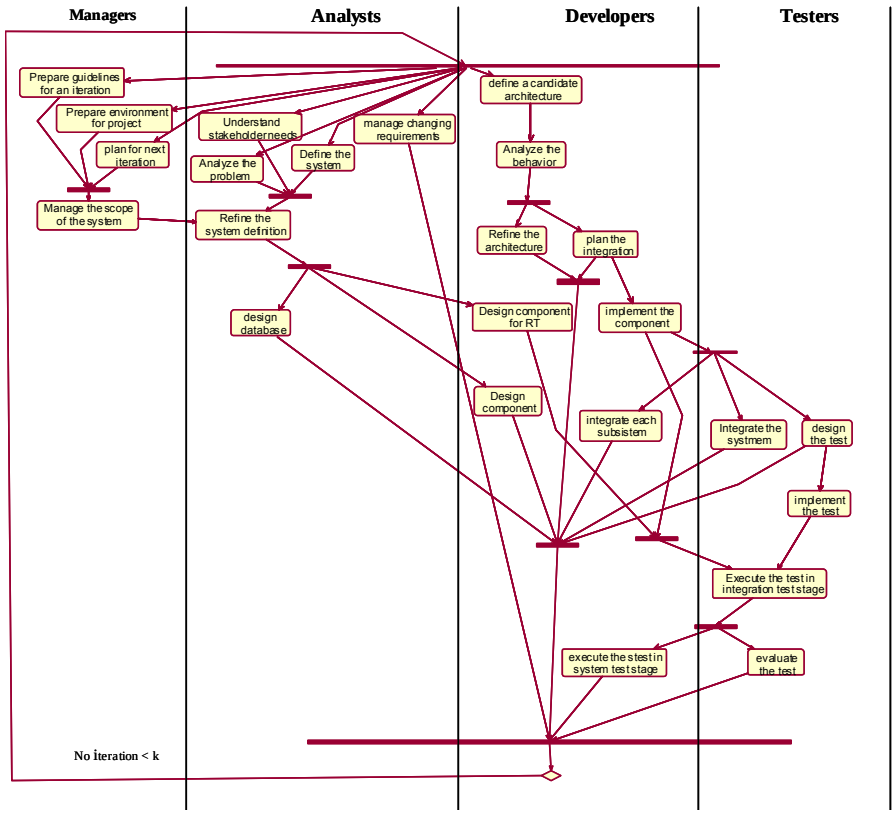

Fig. 2. Activity Diagram relatives to the Elaboration Phase

1. Manager*: Phases description*

For every phase of the RUP process (Inception, Elaboration Construction, and Transition) the manager, with reference to the UML static model described in the previous section, has to specify the flow of the involved activities using the Activity Diagrams (ADs). In each of these diagrams, the decisions and conditions as well as the parallel execution of the activities must be shown. In particular the manager should decide: to possibly suppress some of the RUP activities of entire workflows according to the specific development exigencies of his/her organization, or to specify how many interactions must be performed for each phase. In Figure 2 we report an example of a developed AD. In this case the activities of the different workflow, every one associated to a representative SD, are considered as a sort of "building bricks" that the manager fits into the activity diagrams for describing the overall structure of the development process.

2. Manager*: Organization description and Roles specification*

In this step the manager must describe in a Deployment Diagram, DD, the organization structure. Its nodes can refer to both classical resources (device, processor, database) or people team. In particular the DD models also the communication nodes: for instance, the Intranet to access a common database, or a meeting room symbolizing a "communication channel" among different teams. The Figure 3 is an example of an annotated DD.

In this step the manager must also specify the associations between roles and personnel. The RUP modeling supplies the manager with a Class Diagram with the roles specialization; therefore he/she has only to reorganize the association between the different classes accordingly to the organization exigencies. For example in Figure 4 the class "designer" is associate to a real person (therefore it becomes a superclass) who can assume also the roles of Design reviewer, Database designer and so on (the subclasses)

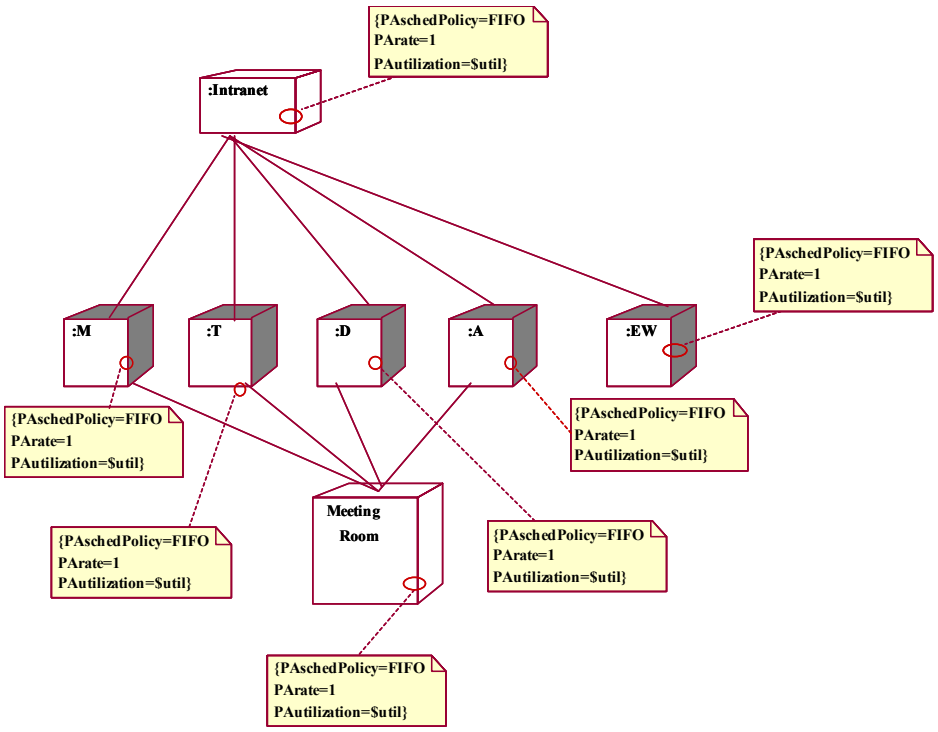

Fig. 3. Deployment Diagram

3. Manager*: SDs annotation*

In this step the manager has to better specify the activities belonging to each Activity Diagrams developed in Step 1. Every activity is in fact associated to an annotated SD, therefore the manager has only to refine the parameters or values of the stereotypes of the SDs description. The Figure 1 is an example of an annotated SD.

4. Automatic*: SPE model generation*

Due to space limits we do not report in this paper the detail of the SPE model generation, more detail can be found in [2] method naturally suits to the RT-UML diagrams and derives automatically: a model for the planned activity by obtaining for each involved SD and Activity Diagram a corresponding EG (the SM based on EG), and a model for the involved teams (the MM based on EQNM).

5. Automatic*: Model evaluation*

The EQNM obtained in the previous step and representing both the teams and the activities can be solved to obtain results such as the completion time for the whole project (or for a single phase) and the resources utilization (see Section 4).

6. Manager*: Analysis of results*

The results obtained in step 4 are analyzed and, if different from those expected (or desired), the manager can go back to the previous steps (1, 2, 3 or 4 according to the obtained results), make some modifications to problem settings and repeat the process.

3 An Example of Propean Application

To see how Propean works, we will apply it here to a case study consisting of a hypothetical project development. Although this example is built ad hoc for illustrative purposes, its organization and the assigned parameters (people involved and planned time for the composing steps) reflect faithfully the management practices of a real world organization. We describe here the project and in the next section provide the results obtained by applying Propean to it.

The system to be developed is composed of two large subsystems, A and B. Subsystem A consists of three components, and subsystem B of two components. For clarity, the development process of this system is represented in Fig. 5 by means of an Activity Diagram.

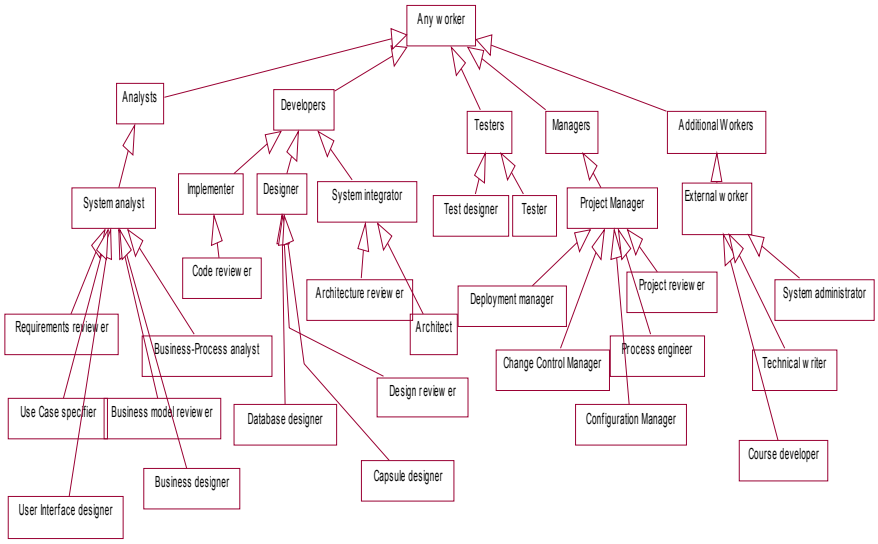

Fig. 4. Class diagram relative to the Roles distribution

To apply Propean, we have modeled the process parts relative to every node of this AD using the "bricks" methodology overviewed in the previous section. Each node is detailed into the RUP activities referring to it and modeled by means of one or more SDs. The various activities corresponding to the nodes of this AD are distributed among the four phases of RUP, and associated with the activities forming the various workflows. For instance, considering the AD in Fig. 2, the activity "Define the system" in the Analysts swimlane, implements the activity "System Design" of the AD of Fig. 5. In other words, we tailored the generic RUP model in Propean to the specific needs of this project, that is a small one, and therefore we obtained a rather simplified RUP configuration. Then, according to step 3 we have annotated the SDs with the estimated duration of each activity. The estimations were made by an industrial manager as an average "guess" based on ERI standard parameters, assuming: the size (in terms of code lines) of the components as small, a medium system complexity, the existence of a design basis, a not new technology, and medium competence of project team. Moreover, we assigned the personnel who will have to carry on the planned activities. Specifically, we assumed as the initial configuration for analysis the following, referred to as Conf_I:

- 1 Project Manager (PM)
- 1 System Analyst (SA)
- 1 Designer (D)
- 1 Implementer (I)
- 1 System Integrator (SI)
- 1 Test Designer (TD)
- 1 Tester (T)
- 1 External Worker (EW)

Note that this is the initial configuration; during the analysis with Propean it may be the case that the hypothesized configuration shows itself as not appropriate and alternative configurations are found as more effective. This is in fact one of the objectives of Propean, to assess whether the people utilization is adequate with respect to project needs.

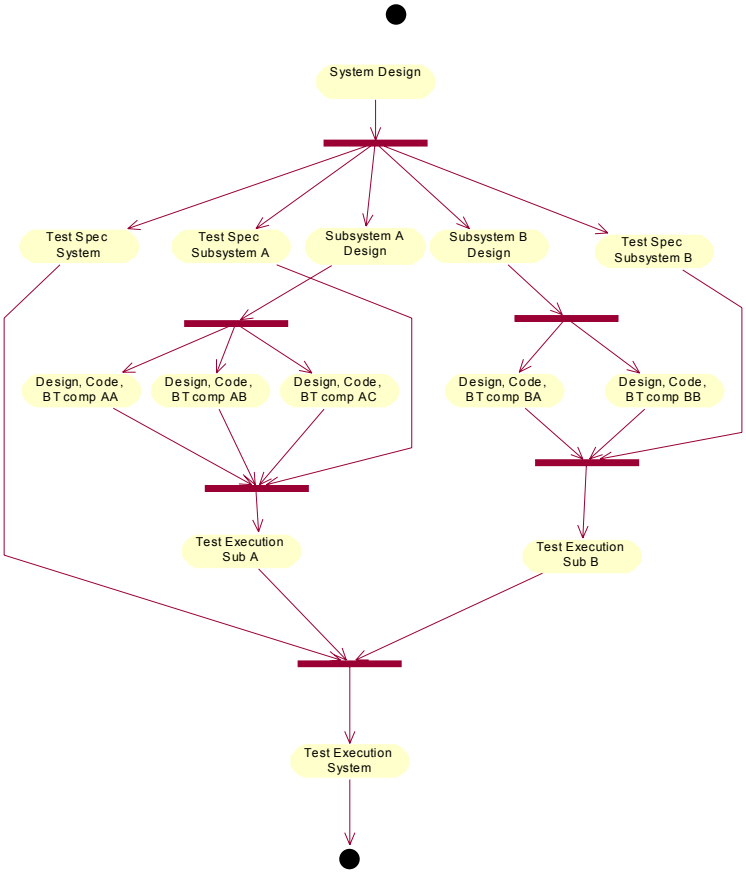

Fig. 5. Development process representation

4 Analysis of Results

We show now and comment some of the estimations that can be automatically obtained by means of Propean.

At a first run, we derived the expected time to completion for the initial configuration Conf_I shown above. The estimated times for the four RUP phases considering the development of the system stand alone, i.e., without any concurrent project development which could compete with this for resources, are given in the first column of Table 1. Conf_I results are given in the white part of the table.

But the real potential of Propean is that we can also make estimations in the more realistic hypothesis that people are not dedicated full time to this project, and that other RUP processes are going on concurrently. For instance, in the table, we also show the estimated time to completion of each process, assuming that two or three processes are concurrently running; we can also make hypotheses on how the processes are interleaved. So for instance we considered the case that the 2 or 3 processes started contemporaneously (0 days of displacement), or that a process starts after 30 days the preceding one started (30 days of displacement). The times are given in days for one process.

As plausible, we obtain that the concurrency among more projects brings large delay on the schedule: a project alone takes 188 days to complete, but if there are two projects to manage contemporaneously, it will take almost 90 days more; this delay is a little diminished if some displacement is inserted between them, so that people can finish one task on a project before being engaged in the other one; but as the Elaboration phase in this project is very long, then we see that the advantage is not big and involves the first period of the project. Things get worse the more projects in parallel we add. The other interesting feature of Propean is that we can look at how the involved teams are charged, and identify the bottlenecks in people configuration.

Tab. 1. Propean estimated times (in days) for the RUP phases

		Number of Concurrent Projects				
	Phases	1	2 (0 days displ.)	2 (30 days displ.)	3 (0 days displ.)	3 (30 days displ.)
Conf_I (8 pp)	Inception	28,81	44,72	40,25	57,63	54,33
1 PM, 1 SA	Elaboration	104,58	155,76	141,35	194,21	187,23
1 D, 1 I, 1 SI	Construction	29,56	35,54	35,79	46	44,32
1 TD, 1 T	Transition	25,33	38,30	36,68	49,5	48,61
1 EW	Total	188,54	275,14	254,62	349,67	337,25
Conf_A (8 pp)	Inception	28,89	43,42	40,18	56,58	50,38
1 PM, 1 SA	Elaboration	91,88	127,06	120,24	156,63	142,82
2 D, 1 I, 0 SI	Construction	25,73	33,36	33,26	42,04	40,97
2 TD, 0 T	Transition	25,88	37,96	36,53	50,16	47,08
1 EW	Total	172,53	241,49	230,62	308,02	283,07
Conf_B (6 pp)	Inception	28,81	43,1	40,08	56,94	50,84
1 PM, 1 SA	Elaboration	104,58	152,57	142,96	190,43	179,26
1 D, 1 I, 0 SI	Construction	29,56	37,72	36,03	47,85	45,46
1 TD, 0 T	Transition	26,18	18,14	36,26	49,5	47,79
1 EW	Total	189,38	271,41	255,88	349,47	325,02

Tab. 2. Propean estimated utilization rate of people for the RUP phases

	Number of Concurrent Projects				
Personnel	1	2 (0 days displ.)	2 (30 days displ.)	3 (0 days displ.)	3 (30 days displ.)
PM	0,09	0,13	0,13	0,15	0,15
SA	0,33	0,45	0,44	0,55	0,52
D	0,44	0,6	0,59	0,71	0,68
I	0,31	0,44	0,42	0,55	0,51
SI	0,005	0,006	0,007	0,009	0,008
TD	0,25	0,34	0,33	0,4	0,38
T	0,06	0,09	0,09	0,11	0,1
EW	0,13	0,17	0,16	0,2	0,19
PM	0,10	0,15	0,14	0,18	0,17
SA	0,36	0,52	0,47	0,62	0,59
D	0,24	0,35	0,33	0,4	0,4
D	0,24	0,34	0,32	0,43	0,39
I	0,340	0,49	0,47	0,58	0,57
TD	0,16	0,24	0,22	0,28	0,28
TD	0,16	0,24	0,23	0,29	0,27
EW	0,13	0,19	0,18	0,23	0,22
PM	0,10	0,14	0,13	0,16	0,15
SA	0,33	0,46	0,43	0,54	0,52
D	0,44	0,62	0,59	0,71	0,69
I	0,310	0,44	0,42	0,53	0,51
TD	0,32	0,44	0,42	0,51	0,5
EW	0,12	0,17	0,16	0,2	0,21

In Table 2, we report the utilization rate of people (in the interval 0-1, where 0 means idle and 1 means the person time is saturated at its maximum). We can see that in the planned workflows, the System Integrator SI and the Tester T are idle most of the time, as their utilization rate is very low compared to the other involved people. Therefore, we analyze how the RUP performance would change if we think to assign different activities to these two people: in the light gray part of the two tables we report the results obtained assuming a different configuration, called Conf_A, in which the person who before acted as a System Integrator is now assigned part of the activities of the Designer, while the person acting in Conf_I as a Tester is here given a Test Designer role. As obvious, we are trying also to consider the expertise of people, and we are reconfiguring them accordingly. The reconfiguration allows the manager to save 16 days in the schedule for the stand alone project, and even more days in the multiproject scenarios. Moreover, if we now look at the utilization rate of people, we see in Table 2 that the effort is more evenly distributed among people.

We have also investigated a different hypothesis, called Conf_B: given that SI and T are not doing much, we move away them from this project, redistributing their tasks to the other people (SI task to D, and T task to TD) i.e., we only assign 6 persons to the process. In this case the results are shown in the dark gray part of the two tables. We still get a more even distribution of effort among people, while the time to completion remains quite similar to that of Conf_I (i.e. we get almost the same time with less cost).

Hence in conclusion the results of Propean analysis for this case study leave the project manager with either of the alternatives: getting the project completed in shorter time and more rational employment of resources with 8 people, or getting it completed in almost the same time of the initial configuration, but employing only 6 people. What appears clear is that the initial configuration is not an effective choice. In addition, Propean provides quite reliable estimates of schedules, based on the manager estimations for the single activities.

5 Related Work

In this section we present a short survey both of the existing methods and tools for project/process management evaluation out of any reference to UML notation, and of the impact of UML introduction in process modeling and evaluation.

5.1 Traditional Methods and Tools

Two crucial aspects of project management are resources distribution and activity planning during the software development. To make a realistic planning, managers need to consider the current workloads of human resources and take the most appropriate decisions for meeting the project deadlines. The decisional support they can use generally is of two kinds. One consists of traditional techniques, like Control Charts or Gantt Charts [3], which visualize resources and personnel and distribute them among the phases of project development. Tools oftentimes support these methods, which are extremely intuitive, but generally the validity of the plans depends strictly on the subjective skill of the managers. Besides, the use of these techniques in a multiproject context, as we do naturally by Propean, could be rather difficult.

The second kind of decisional support consists of specialized tools for managers, like Microsoft Project tool [15] or the Kerzner Project Management Maturity Online Assessment tool [10]. These provide a valid help for maintaining an updated database of the available people and resources, and for producing and visualizing a project plan. However, most tools consider only a specific aspect of management, focusing for example either on the completion time or on the personnel distribution and, more importantly, they cannot explicitly manage several contemporaneous projects. Considering the distribution of resources in a multiproject environment, PERT (Project Evaluation and Review Technique) [12] and CPM (Critical Path Methods) [5] are probably the first proposed methods. They describe an idealized flow of project activities, in which no new project is introduced over time and activity times are treated as deterministic. Markov chain models [5, 21], which assume activity time exponentially distributed and use matrix methods for deciding the task time order in development [3], were the natural subsequential evolutions.

Finally, the majority of available tools apply ad hoc algorithms for simulating the project evolution, based on some parameters values introduced by the user. Some of those tools generate approximate predictions without any guarantee of statistical significance. In contrast, in Propean the predictions rely on the proven sound statistical techniques of performance engineering.

5.2 UML Based Approaches

The widespread use of UML has convinced several authors to consider/apply it also as a process modeling language (PML). Apart from its popularity, UML presents some attractive features as a PML: it contains a number of diagrams that are appropriate for structural and behavioral process modeling, it is extensible, supported by tools and provides a standard textual output. Moreover, it bears the implicit advantage that process model definition can be communicated easily to a large number of people.

Several approaches have recently been proposed [6, 7, 8, 9, 14], which provide guidelines to use the diagrams/constructs already available in UML, as well as to extend them for process description. In these approaches UML is used as an intuitive and rich linguistic support, and is exploited mainly as a "universally understood" language. The work in both [6] and [9] explores the possibility of using UML as a PML for process enactment. Specifically, [9] provides a UML-based process model definitions and describes how the selected models (class, state and collaboration diagrams) can be formalized through graph based transformation into enactable description. Similarly, the work presented in [6] suggests a process descriptions by means of a subset of UML models (i.e., class, activity and state diagrams), proposes a formalization of the semantics of this subset of UML diagrams and presents the translation of UML process models into code, which can be enacted in a specific process-centered environment. We share with the above work the need to model a development process by UML, but the goals are different: while the above authors want to model a process with UML for enacting it, we model a process to evaluate its performance. To the best of our knowledge, ours is the first research to employ a UML-based approach to support manager's decisions, applying SPE techniques.

Finally, in our previous works [1, 2] we explore the possibility of using UML to model a waterfall development process [1] or a specific project phase (the release decision after testing) [2] with the goal of producing models allowing estimates of schedule and resource utilization. Here we have generalized those result to the whole RUP model.

6 Conclusions and Future Work

Our research is aimed at introducing reliable performance-based measurement practices in project management. We have presented the Propean methodology for assisting decision makers within RUP. By modeling in RT UML a RUP process tailored to the needs of a specific organization and a specific project, and relying on classical SPE techniques, we can obtain predictions about the time to completion of the RUP phases and the utilization of the involved teams. For space limitations, we could only provide here a quick overview of what is the methodology, and focused rather on how it could be exploited by managers.

The methodology is intended to be of help for managers in taking decisions about how to allocate the people to a process, and reliably predict process schedules. Notably, Propean adoption within an organization aims at a *strategic planning*, it is not to be seen as an alternative technique to plan a single project, in which case it could still be used, but perhaps would be oversized.

The focus of Propean is on multiproject planning and scenario building. Existing techniques for multiproject planning (i.e. Critical Chain) even though capable to manage a multiproject environment, do not allow managers to build and simulate different scenarios.

One major objection that we foresee to the technique is its complexity; however, on one side it reflects the nature of analyzed subjects: to optimize multiproject environment planning and to effectively produce process roadmaps in the medium/long term is not an easy job. On the other side, we believe that in the same way that RUP is a very detailed framework that is

packaged as a product and tool-supported for easy adoption, Propean itself could be pre-developed into a package tool-supported that already provides the generic pre-built RT UML diagrams of RUP. The manager task would be to tailor the existing diagrams and assign values to the RT UML parameters. We have already started the construction of this generic Propean framework, and our future work will be to refine and make available it for easy adoption. Of course, we will continue Propean validation in collaboration with industries.

In a future perspective, the contribution of Propean is that the process knowledge and process adherence to its RUP model becomes really a part of an adopting enterprise culture. Propean is not bureaucracy, it is not an off-line document living by its own, but it should become the basis of an organization way of thinking and culture.

7 REFERENCES

1. Basanieri, F., Bertolino, A., Marchetti, E., Mirandola, R.: Automating the Management of Teams and Tasks in Software Multiprojects using UML and Queueing Networks. To appear in Proc. SNPD02, Madrid, Spain (June 2002)
2. Bertolino, A., Marchetti, E., Mirandola, R.: Real-Time UML-based Performance Engineering to Aid Manager's Decisions in Multi-project Planning. To appear in Proc. WOSP 2002, Rome Italy, (July 2002)
3. Burr, A., Owen, M.: Statistical Method for Software Quality: Using Metrics for Process Improvement. Int. Thomson Computer Press, (1996)
4. Cortellessa, V., Mirandola, R.: Deriving a Queueing Network based Performance Model from UML Diagrams *WOSP2000*, Ottawa Canada, (September 2000), 58-70
5. Dean, B. V.: Project Management: Methods and Studies. North-Holland, Amsterdam (1985)
6. Di Nitto, E., Lavazza, M., Schiavoni, Tracanella, E., Trombetta M.: Deriving executable process descriptions from UML. ICSE 2002, Orlando, Florida, (May 2002)
7. Eriksson H.E., Penker M.: Business Modeling with UML, Wiley Comp. Pub., (2000).
8. Franch, X., Ribo, J. M.: Using UML for modeling the static part of a software process. UML99, October 1999, LNCS 1723, pp. 292-307
9. Jager D., Schleicher A., and Westfechtel B.: Using UML for Software Process Modeling. ESEC/FSE'99, Toulouse, France, LNCS 1687, Springer, (September 1999)
10. Kerzner Project Management On-line at http://www.iil.com/brochures/kerzner.htm
11. Kruchten P.: The Rational Unified Process – An Introduction, Addison-Wesley (2000)
12. Kulkarni, V. G., Adlakha, V.G.: Markov an Morkov-Regenerative PERT Networks *Oper. Res. (1986)* Vol. 34, 769-781
13. Lavenberg S.S.: Computer Performance Modeling Handbook (New York, 1983), Academic Press
14. Marshall C: Enterprise Modeling with UML: Designing Successful Software through Business Analysis, Addison-Wesley, (2000)
15. Microsoft Project On-line at http://www.microsoft.com/office/project/
16. Mirandola R., Cortellessa V.: UML based Performance Modeling of Distributed Systems, *UML2000* (York UK, October 2000) LNCS 1939, Springer Verlag, 2000
17. RUP available at http://www.rational.com/products/rup/index.jsp
18. Selic B.: Response to the OMG RFP for Schedulability, Performance and Time OMG document Ad/2001-06-14.
19. Smith, C.U: Performance Engineering of Software Systems. Addison-Wesley, Reading, (MA, 1990).
20. Smith, C.U.: Performance Engineering of Software Systems. Addison-Wesley, (MA, 2001).
21. Weiss, G.: Stochastic Bounds on Distribution of Optimal Value Function with Application to PERT, Network Flows and Reliability *Oper. Res.* Vol. 36, (1986), 595-605.