

Performance Measures for Supporting Project Manager Decisions

Antonia Bertolino,^{1*,†} Eda Marchetti¹ and Raffaella Mirandola²

¹ ISTI-CNR, Area della Ricerca di Pisa, via Moruzzi, 1, 56124, Pisa (Italy)

² Dipartimento di Elettronica e Informazione-Politecnico di Milano, Via Ponzio 34/5, 20133 Milano



Research Section

Software measurement is a crucial technology along the software development process, and notably for project management. Decision-making in project management requires in fact, accurate estimations and the support of metrics to monitor and control the varying factors affecting the development process.

Managers must make reliable schedule predictions and optimize personnel utilization. They therefore must be able to dynamically evaluate whether the resources assigned to a job are sufficient and whether the organization structure is adequate to meet the scheduled deadlines.

To support managers in these tasks, we propose a method called Propean (for PROject PERFORMANCE ANALYSIS) based on the combination of classical performance engineering techniques and the Unified Modelling Language (UML). The combined application of these disciplines guarantees on one hand, a validated approach for modelling and estimating the 'Quality of Service' parameters when they come to the development process. Performance engineering techniques are in fact based on rigorous mathematical formalisms such as queueing networks, stochastic Petri nets and Markov models. On the other hand, the adoption of UML provides managers with an easy-to-use front-end representation of the process closer to the design notations they routinely employ. To illustrate Propean application, in this article, we model the case of a manager who must decide a release date for a product undergoing the testing phase. Copyright © 2007 John Wiley & Sons, Ltd.

KEY WORDS: product release; project management; real-time UML; software performance engineering

1. INTRODUCTION

Project managers (PMs) are faced with difficult decisions all along the software lifecycle. They are under pressure to judge whether the resources assigned to a specified task are adequate or whether

under the existing organizational schemes the predicted time schedules will be met, or otherwise to take the needed countermeasures. Drawing the most effective decisions is very complex, because the involved processes are complex, and the many influencing factors (both human and technical in kind) are in most cases not easily measurable and are tightly intertwined.

To assist managers in their decisional practice routine, reliable run-time process measurements and estimations are required. For this purpose,

* Correspondence to: Antonia Bertolino, ISTI-CNR, Area della Ricerca di Pisa, via Moruzzi, 1, 56124, Pisa (Italy)

†E-mail: antonia.bertolino@isti.cnr.it



we believe that, bypassing the classical application domain of computer operating behaviour, performance analysis (PA) techniques could play an important role. Our proposal is therefore to apply well-known techniques from the field of computer performance engineering, such as software performance engineering (SPE) (Smith 1990), (Smith and Williams 2001) and queuing networks models (Lavenberg 1983).

PA refers to the vast body of techniques and procedures traditionally used to assess and predict the time dependent behaviour of computer devices and networks. The incorporation of such techniques within the development process is referred to as *performance engineering*. The area has been the subject of extensive research since the 1970s and is today quite a mature discipline, with a variety of modelling approaches and solving techniques available (Kleinrock 1975), (Lavenberg 1983).

This article proposes an unusual application of performance engineering techniques: i.e. as an aid to PMs for decision-making regarding the organization of teams and tasks within the software development process. Indeed, by assimilating project teams into the processing elements of performance models, and development activities into the tasks to be performed by those processing elements within established time intervals, the well-known and sound body of performance engineering techniques can be usefully and quite naturally adapted to tasks of relevance for software managers, such as assessing the time to completion of specified activities, handling personnel multi-tasking over different projects, optimizing the workloads in development cycles, deciding about products release, and similar issues.

The advantages of using PA techniques in project management are many. We highlight the intrinsic capability to handle multiple parallel projects and their mutual interference in schedule and resource usage. Moreover, the obtained predictions rely on a solid mathematical background and have statistical validity.

On the other hand, we are aware that the use of PA techniques requires the ability to master extremely specialized formalisms (Pooley 2000), such as queueing networks (QNs), stochastic Petri nets and Markov models, which might not be in the typical background of PMs. To cope with this problem, our idea is to hide the specialized mathematical notations behind a manager-oriented interface: we

aim at building a setting in which managers develop a model of the flow of activities and task distribution among personnel using familiar notations and tools, and then this model is automatically and transparently translated into a format that is processable by standard PA algorithms. Our approach is called Project Performance Analysis (Propean).

Indeed, a great effort is currently undertaken within the software engineering community for providing performance techniques with front-end representations that are easier to use and closer to the commonly employed design notations. The most prominent example in this regard is the Unified Modelling Language (UML), which is the emerging standard notation in industry for system analysis, design and implementation (Rumbaugh *et al.* 1999), (UML 2.0). On one hand, some authors have proposed *ad hoc* methods for performance modelling of systems by means of UML diagrams [(see articles in (UML 2004), (WOSP 2004), (WOSP 2005) and (Cortellessa and Mirandola 2002)]. On the other, in 2002 the Object Management Group (OMG) has adopted as a standard the UML Profile for Schedulability, Performance and Time (SPT) (UMLTM 2003) [currently a new profile called MARTE (MARTE 2005 is under development for addressing some recognized weaknesses in SPT)]. SPT introduces modelling extensions that can handle timing specifications of services, mechanisms and resources (logical and physical); concurrency and scheduling; software and hardware infrastructures and their mutual mappings. Besides, it allows for the flexible introduction of more specialized notations where necessary.

Propean embraces the trend of using UML as an input modelling notation towards performance models, and is based on the standard UML SPT profile. More precisely, the transformation from the input model relies on an earlier existing method (Cortellessa and Mirandola 2002) that used the standard UML augmented with *ad hoc* annotations. In Propean, we have adapted that method to SPT.

The idea of using performance techniques in project management is not completely new, but applications so far have been limited to the case of a single project at a time, as, for example, in Adler *et al.* (1995), or have been developed to handle specific situations, e.g. for simulating the performance of cooperating maintenance service centres geographically distributed, as in Antoniol *et al.* (2004), and not as a general approach. In



contrast, Propean provides a generic solution that can handle multiple projects and can be applied to any situation and workflow of activities.

It is a goal of this research to carry on several trial applications of Propean and develop the relative SPT input models, so that blueprints for various plausible contexts in project management are already on hand, as aside documentation of the approach. In this perspective, we have developed a reference model in Basanieri *et al.* (2002) concerning a waterfall development process; another scenario is proposed here¹ to support the decision of releasing a product based on the analysis of trouble reports, and a more general application example encompassing the modelling of the whole Rational Unified Process (RUP) (Kruchten 1998) has been studied (Bertolino *et al.*, 2002b).

A cautionary word is needed: performance techniques can provide the necessary analytical support to substantiate decisions, but a manager's expertise remains essential for tuning the input models with the proper parameter values, and for tackling the human and the unpredictable aspects. In brief, we propose PA as an *aid*, and *not as a substitute*, for expert managers.

1.1. Paper Structure

Propean uses the UML SPT profile for performance modelling of project management contexts. UML and SPT are briefly introduced in the next section with special emphasis on the PA profile and the introduced annotation.

For illustration, we consider throughout an application example of a manager who must decide whether a product is ready to be released on the basis of the status of the trouble reports compiled during the test and debug phase. The application example is described in Section 3.

In Section 4, we outline the steps of the proposed Propean methodology and its application to our example. Section 5 presents the architecture of a tool realizing Propean and Section 6 illustrates an analysis of the results obtained using Propean. Related work is overviewed in Section 7, while Section 8 presents conclusions and future work.

2. OVERVIEW OF REQUIRED KNOWLEDGE

The UML modelling language is rapidly becoming the standard notation for analysis, design and implementation of Object-oriented systems. We assume in this article that the reader is familiar with the UML notation, and recall in this section only the main characteristics of those UML diagrams (namely, sequence diagram [SD] and deployment diagram [DD]) directly involved in the Propean methodology (for more information see the UML User Guide (Rumbaugh *et al.* 1999) and UML 2.0 Documentation (UML 2.0)). A SD shows an interaction between system elements, arranged in chronological sequence. In particular, it shows the class instances (objects) participating in the interaction along their 'lifelines' and the stimuli (messages) they exchange arranged in time sequence. SDs do not show the associations among the objects, but can provide specific information about the order in which events occur.

A DD shows the configuration of run-time processing elements and the software components, processes and objects that live on them. It is a graph whose nodes are connected by communication associations. Nodes may contain component instances (indicating that the component lives or runs on the node), and component instances, in turn, may contain objects (indicating that the object is part of the component). The DD can therefore show the mapping of components to processing nodes.

Although UML is generally recognized as a useful tool for modelling the functional characteristics of a system [(e.g. see articles in (Models/UML 2005) and (UML 2004)] historically it had ignored non-functional requirements, such as response time, availability, throughput and bandwidth. These constitute important system features, nowadays often referred to in abstract as the Quality of Service (QoS) characteristics.

By general consensus, the lack of a quantifiable notion of time and resources of UML was felt as 'an impediment to its broader use in the real-time and embedded domain' (Selic 2001). As reported in Selic (2001), in 1999 to cope with the needs from this key area, the Analysis and Design Platform Task Force of the OMG issued an explicit request for proposals

¹ A preliminary version of this article can be found in Bertolino *et al.* (2002a).



(REP)² for a UML domain-specific interpretation (to be fully conforming to the UML standard) capable of dealing with non-functional requirements.

2.1. SPT Profile

In response to the OMG RFP, a working consortium of OMG member companies proposed a UML Profile for SPT profile, which has been recently adopted as an OMG standard profile UMLTM (2002).

Presenting a detailed overview of the SPT profile is out of the scope of this article; we give here only the essential background needed to understand the features we use in the Propean methodology. For major details, we refer to UMLTM (2003).

SPT is not an extension to the UML metamodel, but a set of domain profiles for UML allowing for the construction of models that can be used to make (early in the life cycle) quantitative predictions regarding the characteristics of timeliness, schedulability and performance. In particular, effort has been spent both to enable predictive quantitative analyses (e.g. the ability to determine the schedulability of a planned piece of software or its response time) and to model QoS aspects, such as deadlines and priorities.

The idea underlying the SPT is to import, as annotations in the UML models, the characteristics relative to the target domain viewpoint (performance, real-time, schedulability, concurrency), such that the various (existing and future) analysis techniques can usefully exploit the provided features. Generally, domain viewpoints are not often used in practice because they require high expertise and specialized knowledge. The intent of the SPT profile is to overcome this problem by providing a single unifying framework to encompass the existing analysis methods, still leaving enough flexibility for different specializations. At the core of the profile is the *general resource-modelling* framework, which provides a common model of resources and of their QoS attributes. Then, on the basis of this common framework, more specific sub-profiles are defined, i.e. 'profile packages dedicated to specific aspects and analysis techniques'. Their purpose is to specialize the generic concepts to better represent the needs of a specific domain, i.e. to derive a *conceptual domain model*.

The general resource-modelling framework itself consists of three sub-profiles dealing respectively with resource modelling, concurrency and time-specific concepts. In the next section, we focus in particular on the SPT sub-profile we refer to for Propean, i.e. the PA profile.

2.2. Performance Analysis Profile

The PA profile is specifically designed for capturing the performance requirements and specifying the QoS characteristics or execution parameters. At a high abstraction level, the concepts characterizing the PA profile are scenarios and resources whose meaning is described in Table 1.

The SPT PA profile provides UML extensions to deal with the above notions of scenarios, resources and workloads and the associated attributes (in the following, PA attributes), so as to allow for extensive and wide-ranging performance analyses. In our methodology, we are actually interested only in a small subset of these extensions.

PA scenarios can be modelled following either a collaboration-based approach or an activity-based approach [as in (Petriu and Shen 2002)]. In the tradition of Cortellessa and Mirandola 2002, we take here the former approach, and represent a scenario by an annotated SD. The usage of activity graphs might present some advantages in expression (UMLTM 2003) (modification of the Propean approach to allow use of activity graphs is part of our future plans).

2.3. Annotations Used

We report below, in Tables 2, 3 and 4, a short description of the subset of PA annotations we use in Propean. They concern the workload, the steps and the resources involved in the scenario considered. For each annotation we specify the associated stereotype, the attributes (with their description) and the UML extensions (PA attributes) used for representing these domain concepts [for more detail see (UMLTM 2003)].

The numerical values associated with the PA attributes may have different interpretations; for example, they may represent a fixed value, a variable to be estimated, an average value or a distribution, or else they may be a prediction, a measure or

² OMG document number: ad/99-03-13



Table 1. PA main concepts

PA main concepts	Description
Resources	The servers in a performance model can be active (processing) or passive:
Active resource	Is a device, such as a processor, interface device or storage device, which has processing steps allocated to it by the deployment of the system.
Passive resource	Is a resource protected by an access mechanism (e.g. is a semaphore), which is accessed during the execution of an operation. It may represent either a physical device or a logical protected-access entity.
Step	Is characterized by its mean execution number and the host execution demand and might involve multiple concurrent threads due to forking.
Mean execution number	Is the mean number of times the operation is repeated when executed.
Host execution demand	Is the execution time taken on the host devices.
Scenarios	Are ordered sequences of steps, describing various dynamic situations involving the usage of a specified set of both active and passive resources under specified workloads. It is possible to distinguish between a closed workload and an open workload.
Workloads	Are the load intensity and the required or estimated response times.
Closed workload	Is one in which a fixed number of requests cycles while the scenario is executed.
Open workload	Is one in which the requests arrive at a given (predetermined) rate.
Performance measures	Include: resource utilizations, waiting times, execution demands, and response times. Each of these values can be derived from the system requirements or performance constraints (e.g. response time for a scenario); estimated on the basis of experience or previous knowledge (e.g. execution demand); directly measured or simulated.

Table 2. PA annotations for *closed workload*

PA annotation: Closed workload		Stereotype: « Paclosedload »		
Attribute	Description	Associated PA attribute		
<i>Population</i>	The size of the workload, i.e. the number of jobs involved	PApopulation	Values	Description
<i>Response time</i>	The delay between the instant in which the scenario starts and that in which it is completed	PAresptime	\$Nuser	Number of insertions in the data base
			Msr	The modelled distribution
			Mean \$T_to_release	The mean The variable representing the mean of the distribution

Table 3. PA annotations for *Step*

PA annotation: Step		Stereotype: « Pastep »		
Attribute	Description	Associated PA attribute		
<i>Repetition</i>	The number of times the step is repeated	PArep	Values	Description
Host execution demand	The total execution demand of the step on its host resources, i.e. the service demand necessary for accomplishing the request	PAdemand	Nrep	The variable representing the number of jobs in the scenario
			Req	The execution demand of this step
			Mean Ts	The mean The value of the distribution mean

Table 4. PA annotations for *Resource*

PA annotation: Resource		Stereotype: << Paresource >>		
Attribute	Description	Associated PA attribute		
<i>Utilization</i>	Is the computed utilization of processing resources expressed as a percentage. For a passive resource it is the mean number of concurrent users of the resource.	PAutilization	Values	Description
			\$Util	The variable representing the rate of utilization of the resource.
<i>Scheduling policy</i>	Is the policy that controls the resources, i.e. the rules for assigning the resources to a set of steps (active resource) or the access control policy for handling requests from scenario steps (passive resource). The scheduling policy can be for example, FIFO (first-in-first-out), PS (processor sharing), LIFO (last-in-first-out), and so on.	PASchdPolicy	P	P can be equal to FIFO (first-in-first-out), PS (processor sharing) or LIFO (last-in-first-out) PR (Priority).
<i>Processing rate</i>	Is (only for active resources) the relative speed factor of the processor, expressed as a percentage of some normative processor.	PARate	1	The resource works full time on the assigned job.
<i>Is preemptable</i>	Is (only for active resources) the possibility for the resource to be preemptable or not, once it starts the execution of an action.	PApreemptable	Yes	When the resource can be interrupted during its work.
<i>Throughput</i>	Is the rate at which the resource performs its function.	PAthroughput	X	It is the amount of work provided per unit of time.

a requirement. To model PA value semantics, SPT follows a predefined syntax, whereby it is possible to specify all the desired characteristics (for an example of application see Section 4).

3. APPLICATION EXAMPLE

As stated in the introduction, the objective of this research work is to propose sound, reliable solutions to support the manager's decisional process in multi-project management. The proposed methodology can be useful at any stage of development, as soon as the PM is called to dynamically take the most appropriate decision based on the actual project status and the emerging circumstances. In these situations, PA techniques can help predict the outcomes that will result from the manager's assumptions and to figure out early whether under the current workflow the settled objectives will be met.

In particular, the application example we investigate here concerns the release decision of a software product. The factors that influence this decision can be many, including marketing exigencies, timing constraints, quality requirements or resources

availability. Here we assume that the release follows a test and debug phase, and that the decision is primarily driven by the product quality, measured in terms of found bugs. More precisely, we suppose that, as usual, the testers report each failure found during the test execution in a form, called the *trouble report*, and that the product will be released only after the testing is completed with no trouble report left open (although some trouble reports may be 'deferred' until the next release, see below).

We consider that at the beginning of the test phase the manager faces either of two different situations: (i) considering the actual personnel availability, he/she wants to predict early the expected time for release, (ii) for a fixed release time, agreed early on the basis of customer exigencies, he/she wants to decide the most adequate personnel configuration to respect the time constraints. In this article, we illustrate the use of Propean for pursuing either of (i) the goals and (ii) considering a simplified application example derived from Kaner *et al.* (1999), to which we refer for further detail.

As a first step, we model the organization structure of the company considering the testing stage and the management of reported problems (we disregard the teams not directly involved in



these activities). The organization is composed of a PM, a test team T (1–3 people), a development team D (2–4 people) and the system architects (SA) (1–2 people).

The testers start to execute the planned test cases and every few days (we assume three in this example), they insert the trouble reports in an on-line database, called the *tracking system* (TS), which only the testers and the PM can modify.

At each TS update, the PM analyses the trouble reports and adopts a proper resolution for each reported problem. We consider three possible outcomes from his/her analysis:

- The problem must be fixed. The PM classifies the problem as 'open' and passes it on to the developers. In this application example, for simplicity, we assume no prioritization politics among failures, i.e. all reported problems are assigned the same severity (different priorities could also be handled, but the example would be more complicated).
- The problem can be deferred. The PM chooses to leave the problem in the current version of the product and to do the fix in a subsequent release. The problem is classified as 'deferred'.
- The problem is not recognized as such. From the trouble report analysis, the PM concludes that it is not a real problem because the program was actually supposed to work in that way. The problem is classified as 'as designed'.

The TS update with the problem classification as 'deferred' or 'as designed' by the PM closes the trouble report (at least for this product release). If instead the problem is classified as 'open', further actions must be taken as described below.

On receiving the open problem reports from the PM, the developers first analyse them to check whether they have enough information to fix the problems or need further explanation from the testers about the failure symptoms. In the latter case, the workflow may include an interaction cycle with the testers. Occasionally, the developers may realize that the fix requires a major design change and inform the PM, who may require the intervention of the software architects to modify the design, after which the developers modify the code accordingly.

After every problem fix, the testers have to retest the modified parts of the program (regression test). They hence either classify the problem as 'closed', updating consequently, the associated

trouble report in the TS, or possibly generate further trouble reports containing the new problems found during the test phase.

Given this abstract workflow of the activities and personnel involved, the PM can periodically analyse the status of the TS in order to:

- Case (a) Estimate the expected time at which the product can be released, that is, when the TS contains only problem reports classified as 'deferred' or 'as designed', i.e. there are no remaining 'open' problems;
- Case (b) Derive the most efficient personnel organization for releasing the product within the established time constraints.

In case (a), if the estimated release time is too late, for example, with respect to market exigencies, the PM has to take proper corrective actions. For instance, the PM could increase the number of people involved in the development or in the test phase, or else decide to pursue a later release date. Alternatively, if the involved personnel are handling several projects contemporaneously, the PM could decide to temporarily divert the people from one or more of the concurrent projects to focus on this one. Similar considerations can be made for case (b).

In both situations, it is very important that the PM can base his/her resolution on a reliable estimate, not on a subjective guess, and that he/she can objectively take into account all the likely combinations of events.

This is the purpose of the methodology presented in the following section: we intend to supply the PM with a tool that uses performance-engineering techniques to:

- Predict the release time, also allowing for multi-projects management, i.e. the teams are not dedicated full time to a single project.
- Evidence (by looking at the rate of personnel utilization) the component that represents the bottleneck and is mainly responsible of the release time delay.
- Identify the most convenient team composition in order to ensure that all the projects are released within the deadline agreed with the customer, or within the budget allowance.



4. THE METHODOLOGY

We adopt PA methods for the purpose of handling personnel multi-tasking and of optimizing workloads in software project management. We follow the metaphor that:

- *Project teams* correspond to the *processing resources* in performance models;
- *Project activities* are the *tasks* to be performed within established time intervals.

4.1. Performance Concepts Used

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]). Using the above metaphor, the SM captures the aspects relative to the activity planning, while the MM the ones relative to people (over/under) utilization and distribution.

The SM captures the essential aspects of software behaviour by means of execution graphs (EG). An EG is a graph whose nodes represent software workload components, which are a set of instructions or procedures performing a specific task, and edges which represent transfers of control.

Figure 5 represents an example of EG in which diverse types of nodes can be noticed, such as basic, conditional fork and join nodes. Each node is

weighted by use of a demand vector that represents the resource usage of the node.

The MM models the hardware platform and is based on the QN model (Lavenberg 1983) (Figure 6 is an example). A QN is specified by means of the service centre and their connections. Parameters such as job classes, job routing among centres, scheduling discipline at service centres, service demand at service centres are obtained from information derived by EGs and from knowledge of resource capabilities. Once the QN is completely specified, it can be analysed by use of classical solution techniques [simulation, analytical technique, hybrid simulation (Lavenberg 1983)] to obtain performance indices such as the mean network response time or the utilization index.

4.2. Propean Conceptual Mapping and Basic Steps

In Propean, we apply a method proposed in Cortellessa and Mirandola (2002) for the derivation of performance models based on SPE techniques starting from a set of UML diagrams. Precisely, the SM is derived from a SD, and the MM from a DD. The method then extracts from these diagrams the main factors affecting system performance and combines them to generate a performance model.

In Figure 1 we summarize the mapping of project management concepts into the SPE and UML environments







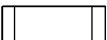



Project View	SPE View		UML View	
	In Soft. Model	In Mach. Model	In Sequence Diagrams	In Deployment Diagram
Project Teams 		QN service centers 	objects 	resource nodes 
Project activities: e.g. Task1	EG Node 	Service demand at QN service centers	object activities requested by messages Task 1 	services provided by resource nodes
information/data exchanged among team	EG Node 	QN service centers 	messages 	communication resources 

Figure 1. Mapping of the project management concepts into the SPE and UML environments



UML environments respectively, highlighting in particular, the information used in the software and MMs.

As seen in this figure, the teams involved in the project development are associated either to the SDs objects or to the DD nodes. The project activities are instead represented by the messages exchanged between SDs objects or the services provided by resource nodes in the DD. Finally, the communications among the teams or the documents exchanged inside the organization are related to either the SDs messages or the communication resources (for example, the intranet device) or links between the DD nodes.

In Figure 2 we show an activity diagram with the steps of the Propean methodology (the two swimlanes, separated by a dashed line, associate each step to the manager and to Propean tool, respectively):

Step 1: Analysis

Description In this step, the PM defines the project activities under consideration. In particular, he/she has to associate to each activity an estimation of the time and the resources necessary for completing it and the role of the people involved. In organizations with stable processes, this information can be derived from previous experience on similar projects.

Example The PM decides to focus the analysis on the testing phase. During this step he/she mainly defines the boundary conditions, such as, for example, the resources involved and the strategy to adopt for project release, and to establish which parameters (symbolic expressions) are relevant for the estimation, possibly postponing their evaluation to steps 5 and 6. We refer to Section 3 for a complete description of this step.

Step 2: Modelling

Description The results of analysis in step 1 have to be modelled as UML diagrams. In particular, the manager should describe, in one or more SDs, the scenario(s) representing the adopted release process. Moreover, the manager should construct a DD modelling the resources available and their characteristics. We note that the manager does not have to repeat this step from scratch each time he/she needs to make estimations about a project. In a mature organization, for similar products, the effort necessary to derive this reference structure will be made only the first time. The same diagrams can then be reused for successive similar applications, by possibly updating the associated parameters, as will be described in the next steps.

Example The SD and the DD representing respectively the sequence of the activities performed during the testing phase and the overall organization of the different teams are presented respectively in Figures 3 and 4 (the meaning of the stereotypes and tagged values will be explained in the next step).

Step 3: Model annotation

Description The two types of diagrams developed in step 2 have to be annotated with the proper values and parameters. The PM should express, by using a comment-based annotation, the attributes associated with events and actions of the diagrams. In particular, referring to the attributes relative to the PA profile described in Section 2.3, the PA attributes of the closed workload will be associated with the first action of the SD; those of the steps will be linked to each of the subsequent message activations; those of the resources will be related to each of the nodes of the DD.

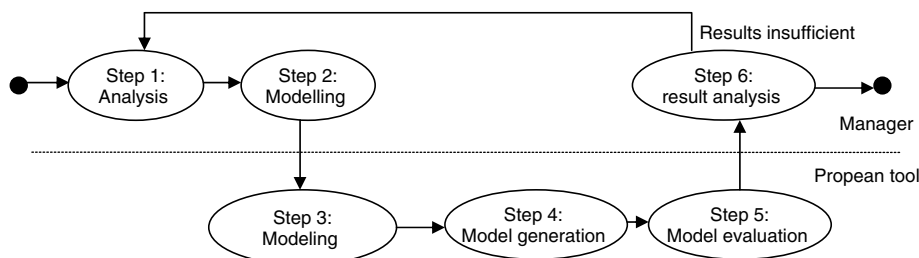


Figure 2. Activity diagram for the Propean methodology

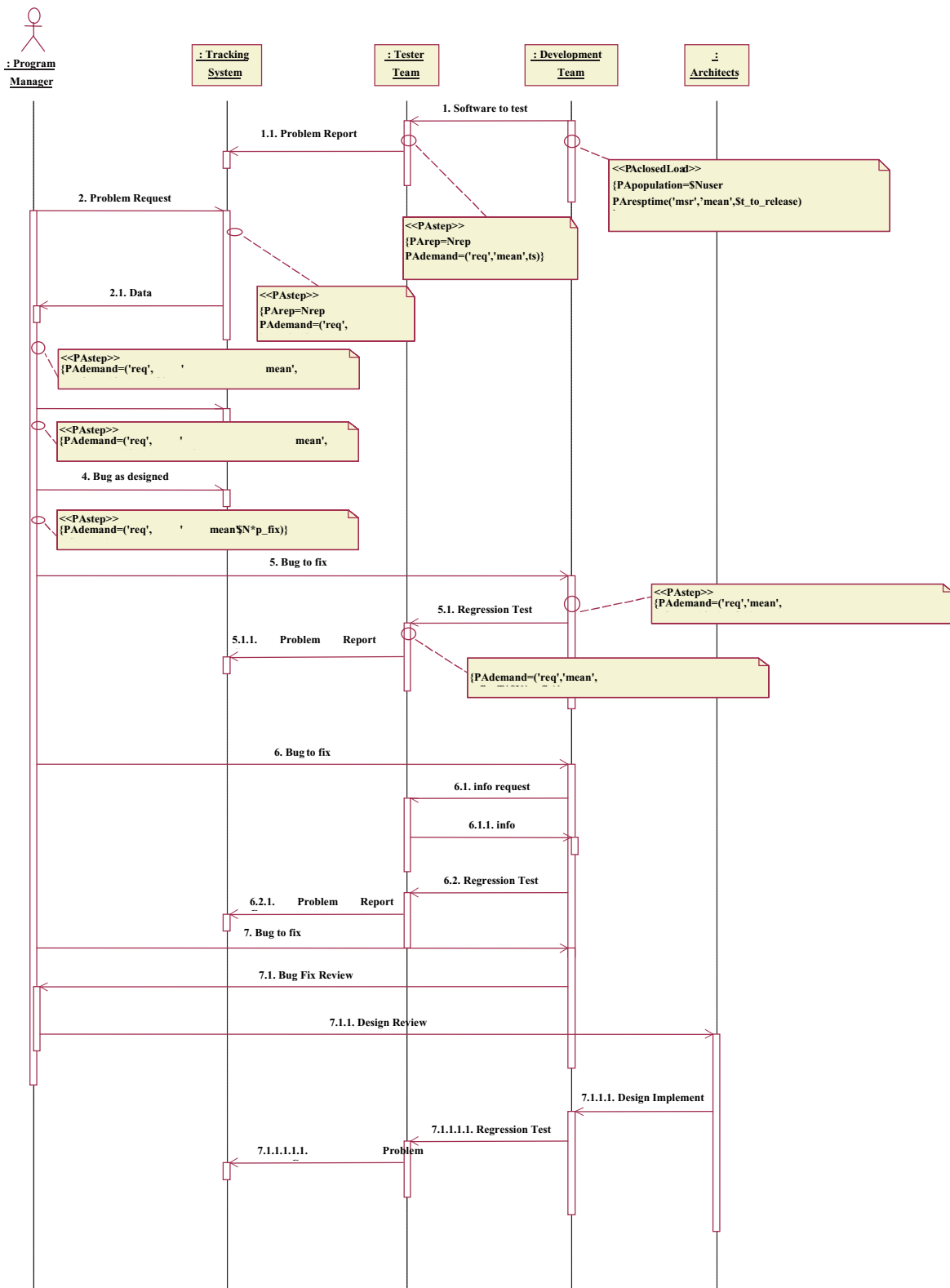


Figure 3. Sequence diagram modelling the workflow of the application example

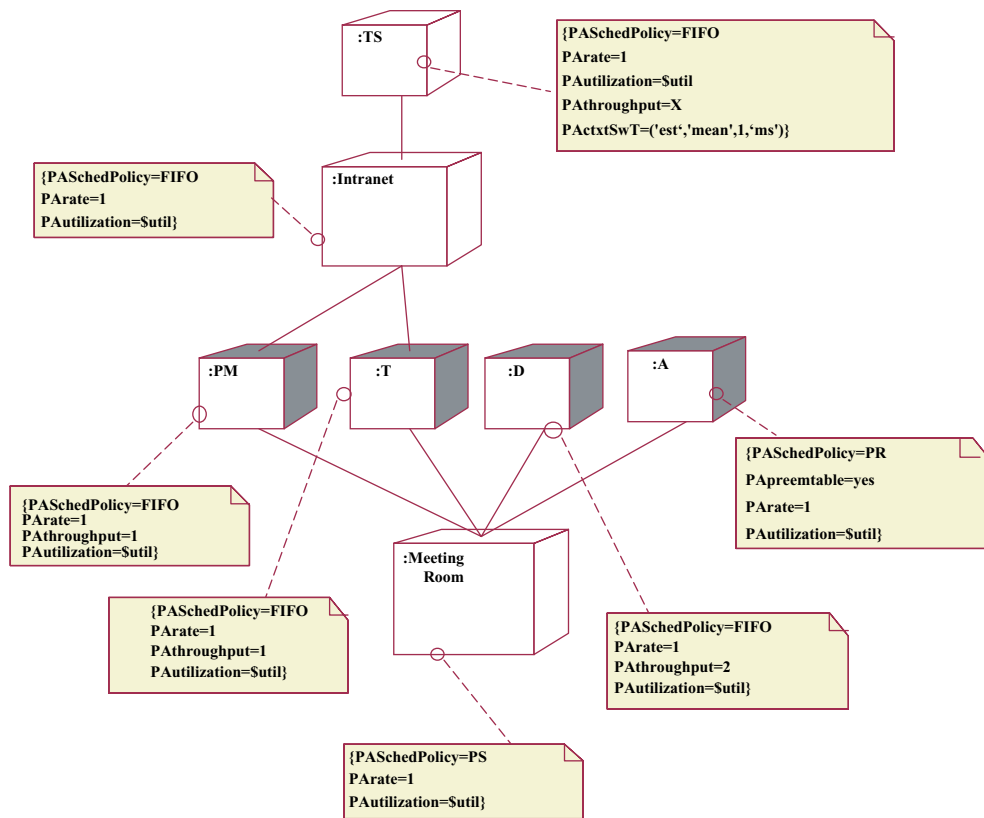


Figure 4. Deployment diagram modelling the 'execution environment' of the application example

Example The annotations used are those described in Section 2.3 and are illustrated in Figures 3 and 4.

Let us consider, for example, the second and third step of the SD ('1.1 Problem Report' and '2. Problem Request' messages, respectively), in which the testers every three days insert in the database a certain number of trouble reports (denoted as N_{rep}). The insertion has a mean value equal to ts . In other words, N_{rep} and ts are the values to be estimated by the PM, possibly with the help of testers.

"They are associated to the PA attributes as follows:

$PA_{rep} = N_{rep}$ i.e. the number of insertions in the data base,

$PA_{demand} = ('req', 'mean', ts)$ where the time necessary ('req') for the testers to insert a trouble report into the database, follows a distribution whose mean is given by ts .

When the PM analyses the trouble reports, he/she observes a variable number of bugs reported ($\$N$). The value to be associated to this variable

generally depends on the project typology and can be estimated for a family of similar products. The Manager can classify each bug as:

- 'open', with probability p_{fix} , spending a t_{fix_PM} amount of his/her time,
- 'deferred' with probability p_{def} , spending a t_{def_PM} amount of his/her time,
- or 'as designed' with probability p_{des} , spending a t_{des_PM} amount of his/her time.

The associated values must be estimated by the PM on the basis of his/her experience. Let us consider for example, the deferred bugs (message '3. Bug Deferred' in the SD of Figure 3) in attribute PA_{demand} the value $t_{def_PM} * \$N * p_{def}$ will represent the time necessary to the PM for dealing with the deferred bugs, where $\$N * p_{def}$ will give the total number of bugs classified as deferred among the $\$N$ reported.

Considering the DD, depending on the resource considered, the meaning of the associated PA attributes refers to 'standard' performance sense



or to Propean logic. For example, in Figure 4, the PATHthroughput annotation of TS node denotes the amount of work provided per unit of time, while the PATHthroughput annotations of nodes PM, T, D, A model the number of people in the team.

Step 4: SPE Models Generation

Description In this step, the Propean tool derives both a model for the planned activity (the SM based on EG) and a model for the involved teams and resources (the MM based on QN).

Example Figure 5 illustrates the EG obtained from Figures 3 and 4, while Figure 6 shows the QN with a team composition made of: one project manager, one software architect, one tester and two developers (1PM, 1SA, 1T, 2D)³.

With respect to the SD and the DD, in this step we have made the following choices:

1. The database TS and the connected Intranet have not been modelled, because the times involved in the TS accesses are orders of magnitude less than the times required by the activity steps (msecs *vs.* days);

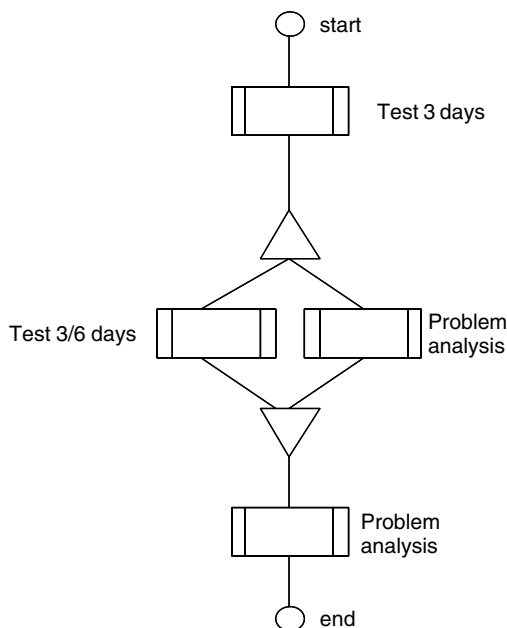


Figure 5. High level EG obtained from SD in Figure 3

³We show these intermediate results for the sake of clarity, while Propean tool works on XML files and directly produces the performance results

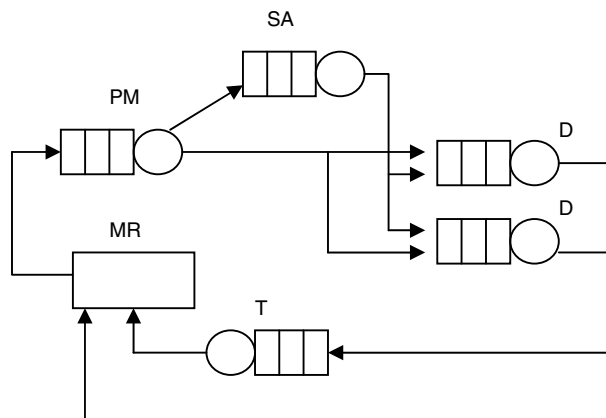


Figure 6. The QN model obtained from SD in Figure 3 and from DD in Figure 4

2. The meeting room has been introduced as a delay centre, modelling the communications with the manager.

Step 5: Model Evaluation

Description In this step, the Propean tool performs a first kind of performance evaluation called *best-case analysis*. The obtained results correspond to the special case of a stand-alone project where the project under study is the only one assigned to the involved project teams (therefore there is no 'resource contention'). Hence, these results provide an optimal bound on the expected performance for each project activity, and can help the manager in identifying a subset of activities that possibly deserve further investigation in the more realistic setting of competition with other projects.

Then the QN obtained in the previous step, which represents the teams and the activities, can be solved to obtain relevant results such as the predicted completion time for the project (or for a single phase), the resource utilization rate, the best resource distribution with respect to a given completion time, and so on.

Example Several analyses can be done by assigning different values to parameters in the QN. Examples of various model evaluations are illustrated in Section 6.

Step 6: Analysis of Results

Description The results automatically obtained in step 5 are analysed by the PM and, if different

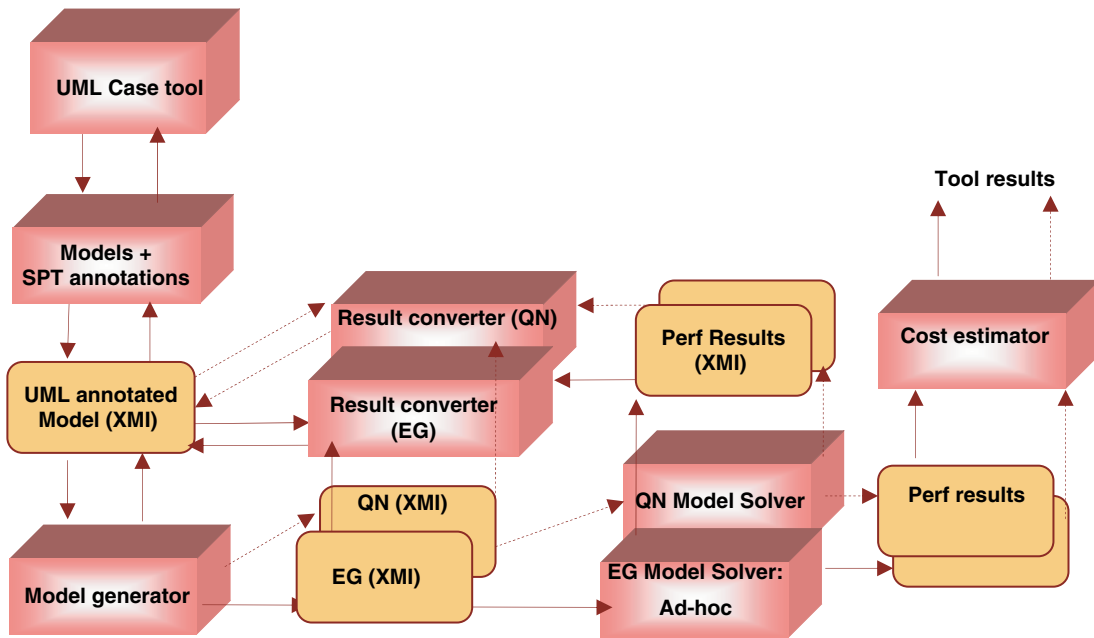


Figure 7. Propean tool architecture

from those expected, he/she can go back to step 1 (or 2), make some modifications to the diagrams or to the assigned parameters, and repeat the process until the desired results are obtained or after a while the unfeasibility of the performance requirements is declared. Clearly, the analysis can produce useful indications about how to modify the parameters.

Achieving the target deadline is certainly important. However, other relevant factors that the manager must consider are the cost and a so-called 'waste' factor (representing the cost of people underutilization) associated to each team configuration selected in step 1. A coarse-grained estimation of cost denoted as $C_G(i)$, $i = 1, \dots, N_c$ (total number of configurations), and of 'waste' factor, denoted as $W_G(i)$ can be computed as follows:

$$C_G(i) = d(i) \times \sum_{k \in \{T, D\}} [(p_k(i) \times c_k) / s_k(i)]$$

$$W_G(i) = d(i) \times \sum_{k \in \{T, D\}} [(p_k(i) \times c_k \times (1 - \rho_k)) / s_k(a)]$$

where:

$d(i)$ denotes the working days for configuration i
 $p_k(i)$ is the number of people in team k
 c_k is the cost associated to each person in team k

$s_k(i)$ is the number of projects shared by team k in configuration i
 ρ_k is the percentage of utilization of team k .

Note that, by use of the attribute PArate we can give also an idea of the expertise of people belonging to the team, with PArate $\in [0,1]$, where 1 denotes a good skilled person, while 0 denotes a beginner. So, we can weight the cost c_k by use of PArate to take into account the cost of different people. Obviously, also the other parameters such as the centre service time modelling people work should be weighted by the PArate value. In the examples above we have supposed, for simplicity, PArate = 1.

Note that currently, owing to restrictions imposed by the PA profile, it is necessary to suppose that all people in the same team have the same skill and cost (an average value can be taken, if this is not the case).

Example The manager can make several kinds of decisions by analysing the results obtained in the previous step. Again, an example of this analysis is illustrated in Section 6.

Discussion It is important to point out that on the manager's side the effort required to employ the methodology is to explicitly derive in an SD a high



level model of the activity workflow and in a DD organization structure. He/she does not need to know all the other details on how such models are then translated into SPE models and then solved (that we described above for completeness).

We understand that even the derivation of the UML diagrams could be felt at first impact as an undesirable extra burden for the already overloaded manager. However, on one hand such diagrams provide a conceptualization and modelling of software processes which are anyhow required in mature organizations; on the other, the kind of models we require objectively should not need much effort: if one has a clear view (as plausibly the manager must have) of how the development process is structured and which are the activities to accomplish and their mutual influences, deriving the UML diagrams that depict them at a high level of detail should not take much labour, especially with the support of an appropriate interactive tool. Besides, we expect that the returns make it worthwhile.

In fact, once such diagrams have been derived, various interesting analyses can be conducted in a completely automated way. The manager can make different assumptions on the parameters of the modelled scenarios and obtain automatically a reliable prediction of what will be the outcomes consequent to each assumption.

5. ARCHITECTURE OF THE PROPEAN TOOL

As already stated, the final goal of this research work is an automated environment that the manager can easily consult in his/her everyday activity to get advice for sound decision-making. With reference to the stepwise procedure described in Section 4.2 and shown in Figure 2, this environment should incorporate a tool fully automating steps 3, 4 and 5, and provide support as well to the other steps pertaining to the manager, facilitating the UML modelling of the workflow and the resources according to the required formats. Currently we have already available some pieces of such a tool (which we used to process the application example presented here), while further implementation work is ongoing to complete the platform. In this section, we overview the architecture of the Propean tool (see Figure 7).

The policy we are pursuing in the implementation is that, where available, we use existing tools and integrate them into the Propean tool. For this reason, to make the SPE calculations, we choose not to implement from scratch a solver, but to transform the UML model into a QN model, to be then processed directly by existing solvers (Smith 1990), (WOSP 2004), (RAQS 2004).

For UML editing, we use the well-known Poseidon tool (Poseidon 2005) augmented through our 'Annotation wizard' to process also the PA annotations. In this way we obtain XML/XMI (XMI 2006), (XML 2006) files that constitute the input to the other modules.

This XMI file enters our *Model Generator* module. This component provides, according to the SPE basic principle, two different models (the two outgoing arcs): a stand-alone EG and a contention-based performance QN.

Let us consider first the round-trip tour for the EG model (continue line). On the basis of the methodology described in Bertolino *et al.* (2002a) and Cortellessa and Mirandola (2002) a global EG is generated from the models of key performance scenarios represented in the SDs (according to their occurrence probability) and its demand vectors are instantiated according to the environment parameters given by the DD. The output of the Model Generator is an XMI document representing the EG.

The developed *EG Solver* module then applies standard graph analysis techniques (Smith 1990) to associate an overall 'cost' to each path in the obtained EG as a function of the cost of each node that belongs to that path. This stand-alone analysis result gives, for each resource, the total average demand, that, in the optimal case, corresponds to the average response time (Lazowska *et al.* 1984).

Let us now consider the second output of the Model Generator module that is the QN model (dashed line). QN model generation is a very complex task that requires determining:

- A. the number of service centres and their characteristics (service discipline, service times),
- B. the network topology and
- C. the jobs in the network with their service demand and routing.

For A and B the necessary information can be derived from the annotated DDs and SDs. For a first assessment, for example, we associate a QN service



centre to each DD node. The service centre scheduling discipline is derived from the PA annotation and the service can be considered exponential with a rate equal to the node throughput (also annotated on the DD). Similarly, the links among the service centres are derived from the ones existing in the DD.

The kind of QN (closed, open or mixed) corresponds to the characteristics of the modelled workload (closed, open and mixed, respectively) annotated on SDs.

Concerning the job characteristics (C) we derive all the essential information from detailed models of key performance scenarios represented in the EGs. The different job classes on the network model the distinct key performance scenarios. The job service demand at the network service centres can be computed from the resource demand vectors of the EG. Similarly, the job routing corresponds to the application dynamics given by the whole EG.

Also for QN we have defined an XMI format to guarantee the interoperability of different tools.

The 'Model Solver' component for the QN is a tool, called *Rapid Analysis of Queueing Systems* (RAQS), developed at the Oklahoma State University (RAQS 2004). An important element in the choice of RAQS was the possibility to describe the input information in textual form.

RAQS allows for the definition and solution of different types of QNs with growing complexity levels.

The *Result Converter* modules receive input both in the application performance goals in terms of PA annotations for the different key performance scenarios and the performance results provided by the EG Solver or QN Solver component. A simple comparison can give insights about the hypothesized project plan and assignment of resources.

When the obtained results fulfil the requirements, they are taken as input by the *cost estimator* module that also performs, for the selected configurations, an estimate of the involved costs taking into account the team (under) utilization.

6. ANALYSIS OF RESULTS

We present in the following two orthogonal applications of Propean to the described application example, illustrating for each of them the parameters to be introduced and the kind of estimations that can be derived. We consider the two situations

in which: (i) the manager wants to predict the completion time of the testing phase (Section 6.1); or (ii) he/she wants to derive the most efficient personnel distribution for completing the testing phase within a fixed time deadline (Section 6.2).

Generally for each diagram several parameters can be varied, depending on the desired prediction. We have considered the following parameters: the estimated duration of the test period, the number of registered trouble reports, the composition of the involved teams and whether they are fully dedicated to the examined project or instead are contemporaneously handling other projects.

6.1. Estimating the Completion Time

Considering the application example described in Section 3, we illustrate some plausible situations. In Propean, each different situation corresponds to a variation in the parameter values in SD and DD. For example, a possible situation could be represented by the following assumptions:

- The planned duration for the test phase of a given product is six days (considering the attribute PArep in the second step of the SD and remembering that we assumed trouble reports are inserted every 3 days, the parameter *Nrep* is set equal to 2);
- For the typology of project under test, on the basis of his/her experience, the manager assumes that the number of trouble reports issued will be equal to 10 (considering the attribute PAdemand in the third step of the SD, the parameter *\$N* is set equal to 10);
- The personnel in charge for the test and debug phase consists of one tester, two developers and one software architect (plus of course, the manager): this configuration is denoted as 1PM, 1SA, 1T, 2D (as illustrated in the considered DD);
- The tester and the two developers are in parallel engaged in another project (the parameter *\$Nuser* of the SD is initialized to 1).

For each parameter configuration, using Propean the manager can thus obtain, early in advance of the testing phase, a prediction of the time in which the product will be ready for release (i.e. no more open trouble reports exist).

We report in Table 5 the obtained results for different parameters values. Note that the Model Solver module of Propean provides results



Table 5. Estimated time to release in days

	Project #bugs	Planned test duration = 3 days				Planned test duration = 6 days				Planned test duration = 9 days			
		T&D Full ded	D Shared 1	T&D Shared 1	T&D Shared 3	T&D Full ded	D Shared 1	T&D Shared 1	T&D Shared 3	T&D Full ded	D Shared 1	T&D Shared 1	T&D Shared 3
1PM 1SA 1T 2D	2	5	6	8	13	8	10	14	25	11	13	17	28
	10	9	10	11	17	12	13	17	29	14	16	20	31
	20	14	15	16	22	16	18	22	32	18	20	24	35

expressed as average values with related confidence interval. In the table the estimated time to release is measured in days, considering one working day equal to 8 hours (optimistic bound), and, for the sake of simplicity, the results are rounded to the closest integer. The table shows the release time when the planned duration for the test and debug phase is 3, 6 or 9 days (with a group of four columns for each case), and when 2, 10, or 20 trouble reports are issued, as indicated in each row.

For each case, then, we derive the estimate when the resources (the people) are fully dedicated to the project under exam (denoted as T&D Full Ded); the test team is fully dedicated, while the developers are handling this and another project (D Shared 1); both the testers and the developers are handling this and another project (T&D Shared 1); and finally both the testers and the developers are handling three more projects in addition to this one (T&D Shared 3).

Going back to the situation described above, in which the test duration was assumed equal to 6 days and the number of trouble reports to 10, the time necessary to complete the testing phase is estimated as 17 days from the start of the test phase (2nd row, 7th column). If the manager was pointing towards a much earlier release deadline, Propean shows it is unlikely that he/she will be able to meet it. Let us assume that the target release deadline is 12 days. We summarize the described situation in Figure 8, where the horizontal marked line represents the target release deadline. Observing in this figure the line of the 'T&D Sheared 1' results, even considering the more optimistic hypothesis that only two bugs are encountered, the release time would be not shorter than 14 days (1st row, 7th column in Table 5). Thus, either the manager can accept a more relaxed deadline, or he/she takes some countermeasure. In particular, if the project under exam has high priority, looking also at the

figure a possible solution could be to take away from the other parallel project the resources (the tester and the developers) that are necessary to complete this one. In this case if they are fully assigned to the completion of the testing phase of this project, then the predicted release time with 10 bugs is reduced to 12 days (2nd row, 5th column), which was the target deadline. Thus, by means of simple SPE analyses, the manager gets statistical predictions that can support his/her decisional process.

On the other hand, adopting the last solution results in an increased project cost, due to the underutilization of certain personnel categories. Another relevant parameter the manager should consider before taking any decision is in fact, the rate of utilization of the involved teams. This analysis is automatically obtained with the parameter assignments used for the estimation of completion time and can be very useful not only

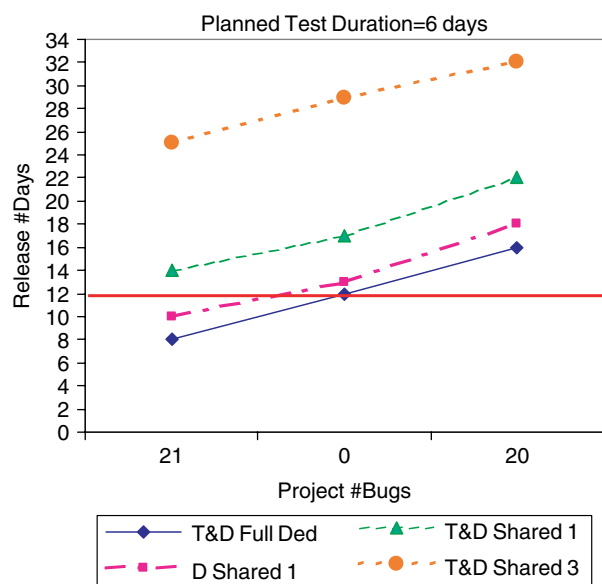


Figure 8. Planned test duration equal to 6 days



Table 6. Percentage of utilization rate of testers and developers

Project #Bugs		Planned test duration = 3 days								Planned test duration = 6 days								Planned test duration = 9 days							
		T&D		D		T&D		T&D		T&D		D		T&D		T&D		T&D		D		T&D		T&D	
		Full		Shared		Shared		Shared		Full		Shared		Shared		Shared		Full		Shared		Shared		Shared	
		ded		1		1		3		ded		1		1		3		ded		1		1		3	
		T	D	T	D	T	D	T	D	T	D	T	D	T	D	T	D	T	D	T	D	T	D	T	D
1PM 1SA 1T 2D	2	21	23	15	59	99	55	100	77	66	10	50	56	99	53	100	76	79	70	65	54	99	52	100	75
	10	29	29	25	57	99	56	100	74	56	29	46	58	99	55	100	76	69	16	59	56	99	54	100	76
	20	31	31	28	55	99	55	100	73	49	30	46	56	99	55	100	74	69	23	56	55	99	55	100	76

to better administrate human resources but also to identify the bottlenecks when a phase takes too long.

In Table 6 we report the percentage of the utilization rate, denoted by ρ , for the tester and the developers considering the same parameters assignment of Table 5. This index is measured by the ratio between the frequency at which requests arrive and the frequency at which the processing element (in our case a team) can deliver services. The utilization rate varies between 0 and 1, where 1 means that the resource is saturated, and can represent a bottleneck; 0 means it is idle, and a good utilization is in the middle.

In the initial configuration we assumed one tester and two developers, employed in this and in another project. We can see that the bottleneck is clearly the tester, as the utilization rate percentage is computed as 99%, while the two developers are well employed, with a rate of 55%. Deciding to fully dedicate one tester and two developers to the test and debug phase allows the manager to meet the deadline, but in such a configuration the developers are underutilized, at 29%.

One further possibility to explore could be to devote one tester at full time, while leaving the two developers on both projects. In such a configuration we would get a release period of 13 days, but the resources are better employed (46% the tester and 58% the two developers).

Analysing the results in Table 5 another interesting fact can be observed: although obviously the duration of the test and debug process can be highly influenced by the number of bugs found, a rational organization of the personnel is more crucial, especially for large enterprises dealing with more development processes in parallel. The release delay in fact, increases faster as the teams get involved in

more contemporaneous projects than if we increase the estimated number of bugs.

For instance, considering a large product with a planned test period of 9 days, when all the resources are fully dedicated, the expected time to release, even foreseeing 20 bugs, is 18 days, against the 20 days estimated to handle half the (i.e. ten) bugs if the tester and the developers are contemporaneously employed in another project. If we further consider a configuration in which the tester and the developers are handling three more projects, even though in this project we optimistically assume to find only two bugs, handling them would take 28 days.

Another possible countermeasure when the predicted release time exceeds the target deadline is to add more personnel to the development. Using Propean, revising the estimates is immediate and again it consists only in changing some of the configuration parameters.

Let us consider, for example, that the personnel in charge of the test and debug phase consists of two testers, two developers and one software architect, plus of course, the PM. This configuration is denoted as PM1, SA1, T2, and D2. We report, in Tables 7 and 8 respectively, the estimated time to release and the utilization rate of the testers and developers.

Considering the initial situation in which the test duration was equal to 6 days, the number of trouble reports to 10 and the committed release time 12 days, even if one more tester is added, the product would be ready in 15 days (as reported in Table 7, 2nd row, 7th column), instead of 17 as with the previous configuration, but this would not be sufficient. In this new configuration, in fact, the manager would be able to meet the target deadline only if the estimated number of bugs is two.



Table 7. Estimated time to release in days

	Project #bugs	Planned test duration = 3 days				Planned test duration = 6 days				Planned test duration = 9 days			
		T&D	D	T&D	T&D	T&D	D	T&D	T&D	T&D	D	T&D	T&D
		Full	Shared	Shared	Shared	Full	Shared	Shared	Shared	Full	Shared	Shared	Shared
		ded	1	1	3	ded	1	1	3	ded	1	1	3
1PM 1SA 2T 2D	2	5	6	6	10	8	10	12	19	11	13	14	22
	10	9	10	11	14	11	13	15	22	14	15	17	24
	20	14	16	16	20	16	17	20	27	18	19	21	28

Table 8. Utilization rate of testers and developers

Project #bugs		Planned test duration = 3 days								Planned test duration = 6 days								Planned test duration = 9 days							
		T&D		D		T&D		T&D		T&D		D		T&D		T&D		T&D		D		T&D		T&D	
		Full ded		Shared 1		Shared 1		Shared 3		Full ded		Shared 1		Shared 1		Shared 3		Full ded		Shared 1		Shared 1		Shared 3	
		T	D	T	D	T	D	T	D	T	D	T	D	T	D	T	D	T	D	T	D	T	D	T	D
1PM 1SA 2T 2D	2	12	24	7	59	55	58	76	77	37	10	26	29	55	55	74	75	40	70	34	54	55	54	70	76
	10	14	29	13	57	55	56	76	75	30	22	24	31	56	56	75	75	36	17	33	56	56	55	72	76
	20	15	30	14	55	54	55	75	70	26	27	22	31	56	57	74	74	33	24	28	55	56	55	74	74

The utilization rate of testers with 10 bugs (Table 8) is equal to 55% (instead of 99% of the previous case). This means that personnel organization in this case is better than before and the tester resource is no more the bottleneck of the development process. As evidenced by the table, in this case assigning the testers and the developers full time to the project, or even only the developers, would be meaningless. Their utilization rates in the two cases (30% and 22%) and (24% and 31%) respectively, in fact, evidences an underutilization of the resources.

Let us now consider also the cost and 'waste' factors and let us compare, for example, the configurations (a) and (b) ($a, b \in \{1, \dots, N_c\}$), where

- (a) corresponds to: 6 days of planned test duration, 10 number of bugs, T&D shared1, 1PM, 1SA, 1T and 2D (Table 5)
- (b) corresponds to: 6 days of planned test duration, 10 number of bugs, T&D shared1, 1PM, 1SA, 2T and 2D. (Table 7)

$$C_G(a) = d(a) \times \sum_{k \in \{T, D\}} [(p_k(a) \times c_k) / s_k(a)]$$

$$= 17[(1 \times c_T) / 2 + (2 \times c_D) / 2]$$

$$C_G(b) = d(b) \times \sum_{k \in \{T, D\}} [(p_k(b) \times c_k) / s_k(b)]$$

$$= 15[(2 \times c_T) / 2 + (2 \times c_D) / 2]$$

Let us suppose, for the sake of simplicity that $c_T = c_D = c$, we obtain:

$$C_G(a) = 17 \times c \times 3/2 = 25.5 \times c$$

$$C_G(b) = 15 \times c \times 2 = 30 \times c$$

As for the waste factor, for configuration a and b above, we can compute

$$W_G(a) = 17 \times [(1 \times c \times (0.01)) + (2 \times c \times 0.45) / 2]$$

$$= 17 \times c \times 0.46 = 7.82 \times c$$

$$W_G(b) = 15 \times [(2 \times c \times (0.44) / 2) + (2 \times c \times 0.44) / 2]$$

$$= 15 \times c \times 0.88 = 13.2 \times c$$

The manager therefore should decide by comparing configuration a and b, whether the increase of the project cost is justified by a reduction of only two days in the completion time and by the increase of waste factor.



6.2. Deriving the Best Personnel Distribution

As in the previous section we discuss some situations for illustration purposes. Considering the application example, we report in this section the results obtained under the following assumptions:

- The planned duration for the test phase of a given product is 3 days (considering the attributes PA_{rep} in the second step of the SD the parameter N_{rep} is set equal to one);
- For the typology of project under test the manager assumes that the number of trouble reports issued will be equal to 2 (considering the attributes PA_{demand} in the third step of the SD, the parameter $\$N$ is set equal to two).

The personnel in charge of the test and debug phase are one software architect, the program manager, and a variable number of testers and developers is variable. The configuration is denoted as 1PM, 1SA, Tt , and Dd where the variables t and d indicate the values to be established by using Propean.

More precisely, the goal of the manager, given the above parameter assignment, is to define the values to assign to the variables t and d , i.e. the best personnel assignment, so that the project can be released within no more than 7 days (considering one working day equal to 8 hours, and the results rounded to the closest integer).

Tables 9 and 10 reports some of the obtained results when the planned duration for the test and debug phase is 3 days, and considering the testers and developers are fully dedicated to the project under exam (denoted as T&D Full Ded); the testers

Table 9. Estimated completion time at various configurations

Configuration 1PM, 1SA, Tt , Dd		Planned test duration = 3 days			
		T&D Full ded	D Shared 1	T&D Shared 1	T&D Shared 3
#Bug 2	$t = 1, d = 2$	5	6	8	13
	$t = 2, d = 2$	5	6	7	10
	$t = 3, d = 4$	5	5	6	8
#Bug 10	$t = 1, d = 2$	9	10	11	17
	$t = 2, d = 2$	9	10	11	14
	$t = 3, d = 4$	9	9	10	12
#Bug 20	$t = 1, d = 2$	14	15	16	22
	$t = 2, d = 2$	14	15	16	20
	$t = 3, d = 4$	14	15	15	17

Table 10. Utilization rate (%) of testers and developers

Configuration 1PM, 1SA, Tt , Dd		Planned test duration = 3 days							
		T&D Full ded		D Shared 1		T&D Shared 1		T&D Shared 3	
		T	D	T	D	T	D	T	D
#Bug 2	$t = 1, d = 2$	24	24	15	60	99	56	100	76
	$t = 2, d = 2$	12	24	0.7	60	55	58	76	77
	$t = 3, d = 4$	0.7	12	0.6	33	38	30	60	52
#Bug 10	$t = 1, d = 2$	30	29	25	57	99	55	100	74
	$t = 2, d = 2$	15	29	12	57	55	56	76	75
	$t = 3, d = 4$	10	15	0.9	30	38	30	60	50
#Bug 20	$t = 1, d = 2$	31	30	27	55	99	55	100	75
	$t = 2, d = 2$	16	30	14	54	55	56	76	76
	$t = 3, d = 4$	10	15	0.9	30	38	30	61	50

fully dedicated, while the developers are handling this and another project (D Shared 1); both the testers and the developers are handling this and another project (T&D Shared 1), and finally both the testers and the developers are handling three more projects in addition to this one (T&D Shared 3).

In particular we suppose that when a team (resource) receives the request for a job, the task is performed by only one of the people available at that moment.

Therefore if the estimated number of trouble reports is equal to 2 and the target release date is 7 days, from the analysis of Tables 9 and 10, a good configuration could be a tester and two developers completely dedicated to this project, one tester and two developers with developers handling this and another projects, or, alternatively, two testers and two developers engaged at the same time in another project. The other configurations can be immediately excluded because of being beyond the deadline.

An alternative situation is that when a team (resource) receives the request for a job, the task is performed by all the people available at that moment. This means that if two people are available, they will work in parallel to complete the job.

Tables 11 and 12 report the estimated completion time for the different configurations and the percentage of utilization rate of the testers and developers. The different policy of job completion is incorporated by the resulting utilization rates of the testers and developers.



Table 11. Estimated completion time at various configurations

Configuration 1PM, 1SA, Tt, Dd		Planned test duration = 3 days			
		T&D	D	T&D	T&D
		Full ded	Shared 1	Shared 1	Shared 3
#Bug 2	$t = 1, d = 2$	5	7	8	15
	$t = 2, d = 2$	4	6	8	14
	$t = 3, d = 4$	4	5	6	8
#Bug 10	$t = 1, d = 2$	7	10	11	17
	$t = 2, d = 2$	7	9	11	16
	$t = 3, d = 4$	6	7	8	10
#Bug 20	$t = 1, d = 2$	12	14	15	21
	$t = 2, d = 2$	11	12	13	19
	$t = 3, d = 4$	9	10	10	13

Table 12. Utilization rate of testers and developers

Configuration 1PM, 1SA, Tt, Dd		Planned test duration = 3 days							
		T&D		D		T&D		T&D	
		Full ded		Shared 1		Shared 1		Shared 3	
		T	D	T	D	T	D	T	D
#Bug 2	$t = 1, d = 2$	26	28	13	59	99	56	100	77
	$t = 2, d = 2$	14	29	0.6	60	52	57	76	77
	$t = 3, d = 4$	11	18	0.6	35	38	32	61	53
#Bug 10	$t = 1, d = 2$	34	35	26	62	99	60	100	78
	$t = 2, d = 2$	19	38	13	64	56	60	77	78
	$t = 3, d = 4$	16	25	14	40	42	38	65	56
#Bug 20	$t = 1, d = 2$	36	37	31	62	99	61	100	78
	$t = 2, d = 2$	20	40	17	63	56	62	77	78
	$t = 3, d = 4$	17	26	16	40	42	39	63	54

7. RELATED WORK

Voluminous literature about project management and development can be found, but little of it treats the problem of multi-project planning and people multi-tasking on several parallel projects. We report here a brief survey of previous related studies and of the more widespread decisional tools.

7.1. Related Studies

Two crucial aspects of project management during development are resource distribution and activity planning. These issues belong to a more general research field, that is, concurrent engineering (CE) (Smith 1997). This discipline became popular with the studies of Imai *et al.* (1985) and Takeuchi and

Nonaka (1986), and has greatly influenced both the academic and the industrial approaches to production. However, these works focus mainly on organizing the tasks within a single project, taking into account the decomposition of a complex product design into smaller activities and their subsequent coordination.

Considering the distribution of resources in a multi-project environment, project evaluation and review technique (PERT) and critical path methods (CPM)(Dean 1985) 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 durations are treated as deterministic. Markov chain models (Krishnan and Ulrich 2001; Weiss 1986), which assume an activity time exponentially distributed and use matrix methods for deciding the task time order in development were the natural subsequent evolutions. In Lee and Miller (2004), a formal simulation model is built to study the dynamics of software multi-project management. Cohen (Cohen *et al.* 2005), based on scheduling theory, proposes a methodology to determine optimal loading of multi-project systems. Different approaches based on mixed-integer optimization to solve planning and design process are surveyed in Kallrath (2005).

The work presented here is close to Adler *et al.*'s (1995). These authors in fact, study the problem of personnel organization and resources distribution among several projects developed at the same time, and like us use QNs and stochastic processing network models to represent product development and identify the bottlenecks in task scheduling. The authors, however, focus on five basic process elements: jobs, tasks, procedure constraints, resources, and flow management control. In particular, a single process may need to handle a variety of job types, which in turn are divided into tasks (i.e. activities or operations). Tasks are connected by precedence relations. The resources are engineers and technicians, who are the units that execute the tasks. The flow management control represents how the resources executed a job's constituent tasks. Loch (Loch 1998) identifies a sixth element consisting of the assessment of individual contributions.

Recently, queuing theory has been applied to model software maintenance requests (Ramaswamy 2000) and to management planning (Antoniol *et al.* 2004). Specifically, in the latter, a queueing approach is presented for staffing process management and



evaluating service levels. The nodes of a multi-stage, multi-centre queueing model are associated with the different maintenance phases. Each stage is considered in series and each entering request is subjected to a sequence of activities before leaving the system.

7.2. Decisional Tools

The decisional support managers can use generally is of two kinds. One consists in techniques or methods that visualize resources and personnel and distribute them among the phases of project development. Examples are represented by the traditional control charts or gantt charts (Black *et al.* 1990), or design structure matrix (DSM) (Browning 2001), which can display the interactions between different teams with the process activities. Tools may support these methods (AceProject 2004; Celexis 2002), (Mind Tools 2005; KickStart 2005), 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 multi-project context could be rather difficult.

The second kind of decisional support consists of specialized tools for managers. Microsoft Office Enterprise Project Management tool (EPM 2005) and the Kerzner Project Management Maturity On-line Assessment tool (Kerzner 2006) represent some examples of specific tools, which provide a valid help for maintaining an updated database of the available people and resources, and for producing and visualizing a project plan. An interesting overview of existing tools and approaches for project and program management are available at Dmoz (2005), PMSD (2005) and (Krishnan and Ulrich 2001) respectively.

The idea of re-adapting existing tools for management purposes is becoming common also for economic aspects, and some proposals can be found in literature. An example is the work of Dickinson *et al.* (Dickinson *et al.* 2001), which shows how to use dependency matrix in combination with the existing Portfolio tools to support decisional process, analyse the interdependences between projects and combine them together. Another solution is presented in Bori *et al.* (2001) where the authors propose a tool for production management optimization, which uses gantt charts and PERT (Kulkarni

and Adlakha 1986), diagrams for visualizing the obtained results.

However, most existing tools consider only a specific aspect of management, focussing for example, either on the completion time or on the personnel distribution and, more importantly, they cannot explicitly manage several contemporaneous projects. Finally, the majority of available tools apply *ad hoc* algorithms for simulating project evolution, based on some parameter values introduced by the user. Some of those tools generate approximated predictions without any guarantee of statistical significance.

8. CONCLUSIONS

In this article, we have discussed the usefulness of applying classical PA techniques to support the management of people and workflows in software processes. We have introduced the Propean approach as an aid for decision-making substantiated by rigorous statistical predictions of time and efforts. Propean input models are derived from UML diagrams depicting the flow of activities and the structure of the enterprise organization.

We have illustrated in an application example, encompassing the test and debug phase of a development process, how the proposed methodology could be of help to a manager in either of two situations: for establishing a reliable release date (while keeping under control the effort charged to personnel) or for deciding the most effective organization of involved teams when the release date is fixed.

To use the Propean methodology, only limited effort is needed on the manager's side to develop the SD and DD diagrams. All the necessary transformations and desired analyses can then be conducted in an automated way by performance model-solving tools.

In evaluating the required effort for Propean application, it must be considered that the models do not have to be re-derived from scratch at each application of the methodology. It is in fact, plausible to hypothesize that the processes that govern the various phases of development in a mature company are standardized and modelled, and do not change completely at each new project. Therefore, after an initial investment to model the various workflows encompassed by the



development process in use, at each next application of the methodology the manager needs only to tune the parameters, or in the worst case to make some update to the existing diagrams.

While we hope this article has provided an evidence of the potential usefulness of PA in project management, we remain aware of course that no tool can replace the expertise and judgment of good managers. We propose Propean as an assistant and certainly not as a substitute for managers. We claim that, if well used, Propean can alleviate the intellectual burden of trying to account for all potential scenarios and circumstances when making predictions, and can certainly provide a valid contribution to more effective decision-making. The real validity of Propean comes out in complex situations such as the case of multiple parallel projects and of personnel multi-tasking, while in simple cases it could be an oversized method.

Our future work will include: the investigation of further features of the SPT profile, such as, for instance, the usage of activity diagrams instead of the SDs and the development of other case studies with their relative UML input models so that Propean blueprints for various plausible contexts in project management are already available; for instance, we are considering to develop ready Propean models of the whole RUP. Finally, we are still working on the Propean tool to enhance its features and to render its use more appealing for manager.

REFERENCES

- AceProject. 2004. Copyright 2001–2004. Websystems Incon line at <http://www.aceproject.com/index.htm>.
- Adler PS, Mandelbaum A, Nguyen V, Schwerer E. 1995. From project to process management: an empirically based framework for analyzing product development time. *Management Science* **42**: 458–484.
- Antoniol G, Cimitile A, Di Lucca GA, Di Penta M. 2004. Assessing staffing needs for a software maintenance project through queueing simulation. *IEEE Transactions on Software Engineering*, vol. 30, no. 1. IEEE Computer Society 10662 Los Vaqueros Circle: Los Alamitos, CA, 43–58; 90720-1314.
- Basanieri F, Bertolino A, Marchetti E, Mirandola R. 2002. Automating the management of teams and tasks in software Multiprojects using UML and queueing networks. *Proceedings of 3rd ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD*. Madrid, Spain, June.
- Bertolino A, Marchetti E, Mirandola R. 2002a. Real-time UML-based performance engineering to aid manager's decisions in multi-project planning. *Proceedings of Third ACM International Workshop on Software and Performance, WOSP 2002*, Rome, Italy, July.
- Bertolino A, Lombardi G, Marchetti E, Mirandola R. 2002b. Performance analysis of the rational unified«Process Product». *Proceedings of 12-th International Workshop on Software Measurement, IWSM 2002*, Magdeburg, Germany, October.
- Black TA, Fine TA, Sachs TA. *A Method for Systems Design Using Precedence Relationships: An Application to Automotive Brake Systems*. M.I.T. Sloan School of Management: Cambridge, MA, 1990. Working Paper no. 3208.
- Bori S, Lores J, Pascual R, Roures E. 2001. PROMAN, planning production management and control system with intelligent interface and advanced forecast. In *Proceedings of ETFA 2001, 8-th IEEE International Conference on Emerging Technologies and Factory Automation*, Antibes (France), 15–18 October.
- Browning T. 2001. Applying the design structure matrix to system decomposition and integration problems: a review and new direction. *IEEE Transaction on Engineering Management* **48**: 292–306.
- Celexis. 2002. Web based project management software copyright 2002 on line at http://www.celoxis.com/html/home_general.php.
- Cohen I, Golany B, Shtub A. 2005. Managing stochastic, finite capacity, multi-project systems through the cross-entropy methodology. *Annals of Operations Research* **134**: 183–199.
- Cortellessa V, Mirandola R. 2002. PRIMA-UML: a performance validation incremental methodology on early UML diagrams. *Science of Computer Programming* **44**: 101–129, Elsevier Science.
- Dean BV. 1985. *Project Management: Methods and Studies*. North-Holland: Amsterdam.
- Dickinson M, Thornton AC, Graves S. 2001. Technology portfolio management: optimizing interdependent projects over multiple time periods. *IEEE Transaction on Engineering Management* **48**: 518–527.
- Dmoz. 2005. Open Directory Project November 27, 2005 http://dmoz.org/Business/Management/Project_and_Program_Management/.



- EPM. 2005. Microsoft Office Enterprise Project Management Solution Published: October 20, 2003, Updated: November 2, 2005 <http://www.microsoft.com/office/project/prodinfo/epm/overview.mspx>.
- Imai K, Nonaka I, Takeuchi H. 1985. Managing the new product development process: How the Japanese companies learn an unlearn. In *The Uneasy Alliance*, Clark H, Hayes H, Lorenz C (eds). Harvard Business School Press: Boston.
- Kallrath J. 2005. Solving planning and design problems in the process industry using mixed integer and global optimization. *Annals of Operations Research* **140**: 339–373.
- Kaner C, Falk J, Nguyen HQ. 1999. *Testing Computer Software*. John Wiley: Van Nostrand Reinhold.
- Kerzner. 2006. Project Management Maturity Online Assessment on line at http://www.iil.com/project-management_training/project_management_maturity_model.asp.
- KickStart. 2005. Project 1997–2005 © Copyright Experience In Software, Inc On line at <http://www.projectkickstart.com/>.
- Kleinrock L. Queueing Systems, Volume 1: Theory, Wiley Interscience, New York, 197.
- Krishnan V, Ulrich KT. 2001. *Product Development Decisions: A Review of the Literature Management Science*, Vol. 47. 1–21.
- Kruchten P. 1998. *The Rational Unified Process: An Introduction*. Addison-Wesley: Boston.
- Kulkarni VG, Adlakha VG. 1986. Markov–regenerative PERT networks. *Operations Research* **34**: 769–781.
- Lavenberg SS. 1983. *Computer Performance Modelling Handbook*. Academic Press: New York.
- Lazowska ED, Zahorjan J, Graham GS, Sevcik KC. 1984. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models* on line at: <http://www.cs.washington.edu/homes/lazowska/qsp/>.
- Lee B, Miller J. 2004. Multi-project management in software engineering using simulation modelling. *Software Quality Journal* **12**: 59–82.
- Loch CH. 1998. Operations management and reengineering. *European Management Journal* **16**: 306–317.
- MARTE. 2005. *OMG-UML Profile for Modeling and Analysis of Real-Time and Embedded Systems*, OMG Document–realtime/05-02-06 edition, January.
- Mind Tools. 2005. Copyright Ltd, 1995–2005, on line at <http://www.mindtools.com/pages/main/newMN-PPM.htm>.
- Models/UML. 2005. *Proceedings of 8th International Conference on Model Driven Engineering Languages and Systems*, Stuart Kent J (ed.). Half Moon Resort: Montego Bay Jamaica, October 2005.
- Petriu DC, Shen H. 2002. Applying the UML Performance Profile: Graph Grammar-based derivation of LQN models from UML specification. *Proceedings of Performance TOOLS 2002*. Springer Verlag: London, England, April 14–17, LNCS 2324.
- PMSD. 2005. Project Management Software Directory Copyright © 1999–2005, Capterra, Inc on line at <http://www.capterra.com/project-management-solutions>.
- Pooley R. 2000. *Software Engineering and Performance: A Roadmap*. Finkelstein A (ed.). The Future of Software Engineering: 22nd ICSE Limerick, Ireland.
- Poseidon. 2005. available at <http://www.gentleware.com>.
- Ramaswamy R. 2000. How to staff business critical maintenance projects. *IEEE Software* **7**: 90–95.
- RAQS. 2004. Documentation On line at <http://www.okstate.edu/cocim/raqs/>.
- Rumbaugh J, Jacobson I, Booch J. 1999. *The Unified Modelling Language Reference Manual*. Addison Wesley.
- Selic B. 2001. *Response to the OMG RFP for Schedulability, Performance and Time*. Addison-Wesley: Reading, MA, OMG document Ad/2001-06-14.
- Smith R. 1997. The historical roots of concurrent engineering fundamentals. *IEEE Transaction on Engineering Management* **43**: 67–78.
- Smith CU. 1990. *Performance Engineering of Software Systems*, Addison-Wesley, Reading, MA.
- Smith CU, Williams L. 2001. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*. Addison-Wesley: Boston, MA.
- Takeuchi H, Nonaka I. 1986. The new product development game. *Harvard Business Review* **64**: 137–146.
- UML. 2004. *Proceedings of the Seventh International Conference on the Unified Modelling Language*, Jardim Nunes N, Selic B, Rodrigues da Silva A, Toval Alvarez A (eds). Springer Verlag: Lisbon, Portugal, October 10–15, 2004, LNCS LNCS 3297.



UMLTM. 2003. Profile for Schedulability, Performance, and Time Specification. On line at <http://www.omg.org/cgi-bin/doc?ptc/02-03-02.pdf>.

Unified Modelling Language (UML). version 2.0. On-line at <http://www.uml.org/#UML2.0> year 2005.

Weiss G. 1986. Stochastic bounds on distribution of optimal value function with application to PERT, network flows and reliability. *Operations Research* **36**: 595–605.

WOSP. 2004. *Proceedings of Fourth International Workshop on Software and Performance WOSP 2004*. Oracle

Conference Centre Redwood City: Redwood Shores, California, USA, January 14–16, 2004.

WOSP. 2005. *Proceedings of Fifth International Workshop on Software and Performance WOSP 2005*. July 11–14, 2005 Palma de Mallorca, Illes Balears SPAIN.

XMI. 2006. on line at <http://www.omg.org/technology/documents/formal/xmi.htm>.

XML. 2006. on line at <http://www.w3.org/XML/>.