International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:2, 2012

Project Management and Software Development Processes: Integrating PMBOK and OPEN

Maurício Covolan Rosito, Daniel Antonio Callegari and Ricardo Melo Bastos

***Abstract*—**Software organizations are constantly looking for better solutions when designing and using well-defined software processes for the development of their products and services. However, while the technical aspects are virtually easier to arrange, many software development processes lack more support on project management issues. When adopting such processes, an organization needs to apply good project management skills along with technical views provided by those models. This research proposes the definition of a new model that integrates the concepts of PMBOK and those available on the OPEN metamodel, helping not only process integration but also building the steps towards a more comprehensive and automatable model.

***Keywords*—**OPEN metamodel, PMBOK metamodel, Project Management, Software Process

I. Introduction

THE increasing concern on software development issues drives organizations to the adoption of software engineering practices. Some most desirable characteristics include the ability to capture the best practices on software development, a good level of flexibility in order to reach a wide variety of projects, and also good management skills.

Developing software products requires the planning and execution of activities, defined in accordance to the scope of the project, and where it is necessary to deal with both management and technical issues. We must consider the fact that projects are always unique and temporary endeavors. They also have one or more goals, require resources and have a defined sponsor. Besides, all projects involve a great level of uncertainty [1]. Still, most models or guides for project management, such as the PMBOK Guide, do not specifically address software development processes. On the other hand, existing software development processes lack of more project management skills in their models or methodologies.

Project management in a software development environment is defined as the management of people and other resources by a project manager in order to plan, analyze, design, build, test and maintain an information system [2].

M. C. Rosito is with the Pontifical Catholic University of Rio Grande do Sul, 6681 Ipiranga Avenue, 90619-900 Porto Alegre, RS, Brazil (phone: 51-3320-3558; e-mail: mauricio.rosito@acad.pucrs.br).

D. A. Callegari is with the Pontifical Catholic University of Rio Grande do Sul, 6681 Ipiranga Avenue, 90619-900 Porto Alegre, RS, Brazil (phone: 51-3320-3558; e-mail: daniel.callegari@pucrs.br).

R. M. Bastos is with the Pontifical Catholic University of Rio Grande do Sul, 6681 Ipiranga Avenue, 90619-900 Porto Alegre, RS, Brazil (phone: 51-3320-3558; e-mail: bastos@pucrs.br).

In order to fulfill these intents, a project manager needs some kind of support, generally based on a project management methodology which can deal with many singular project variants, responsibilities and tasks. Still, software development processes generally provide just a set of practices that deal with certain activities and workflows related to management. Yet these processes do not adequately address human resources and other kinds of resources such as equipment and involved material.

As pointed out by [2], two important software development processes, Rational Unified Process (RUP) [3] and Object-oriented Process, Environment and Notation (OPEN) [4], respectively, need more support for project management concerns. Both RUP and OPEN help the execution of the so called “best practices for software development”. Despite that, RUP, for instance, does not cover essential project management skills like human management and subcontract management. On the other hand, OPEN presents a set of activities and techniques that address areas such as quality, cost rating and management metrics. Nevertheless both models seem to lack enough support on essential knowledge areas of project management, namely: procurement, communication and human resources.

Past and present works on the literature indicate the importance of using well defined software processes in organizations. Meanwhile, there seems to be not enough work in fulfilling the lack of project management skills in those processes. In order to have a more comprehensive process for management and software development, we need to apply well-known project management skills to the appropriate software development process. As a consequence, we need more research for a solution that can provide a greater level of integration among the concepts and models for these two areas. More than that, the desired solution should allow the development of tools that support the decision making process of an organization through the automation of technical and managerial planning processes.

While the Project Management Book of Knowledge Guide (PMBOK) [5] can provide a managerial perspective of the solution, the technical view may be provided by a software process model such as OPEN. The integration of the PMBOK concepts with other software development process (e.g. RUP) was already addressed in [6]. This research adds to the field by proposing the same approach with the OPEN Process Framework. By analyzing how project management knowledge can help improving current software development processes we can derive new tools to support different levels of automation in the planning and execution of activities inside a software project.

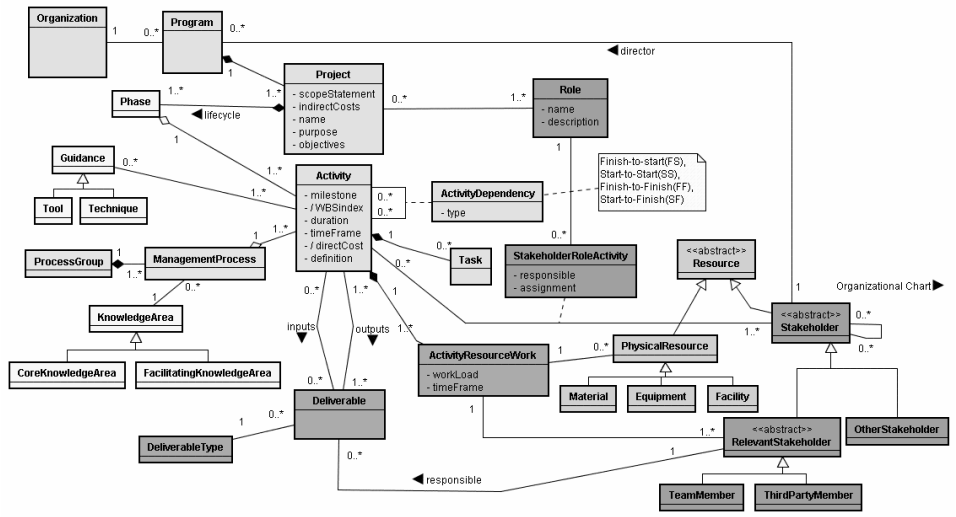

Fig. 1 Project Management metamodel based on the PMBOK Guide [6]

In this text, we first present an overview of the PMBOK and a project management metamodel based on the PMBOK Guide. Section III introduces OPEN Process Framework and the main components of its model. Section IV contains a comparative analysis of PMBOK and OPEN concepts organized as classes. The integrated metamodel for the PMBOK and OPEN is detailed in section V, while the conclusion and comments on future work are presented in section VI.

II. PMBOK Metamodel

A project is a temporary endeavor with the purpose of producing a unique product or service [5]. A project is generally directed to a specific result and involves the coordinated execution of inter-related activities. More than that, projects are planned, executed and controlled by people, and they are constrained by limited resources.

The lack of a methodology for project management, as well as the complexity and volume of the projects in an organization, contributes to an increase of project management problems [7]. But, most management models or guides are not software-specific. In addition, this management models (e.g. PMBOK) are generally more applied to industrial and manufacturing activities. Besides, most of the software development processes generally provide just an adequate set of practices that supports the suggested activities and associated workflows.

According to [5], project management means applying knowledge, skills, tools and techniques to the project's activities, in order to meet or exceed the needs and expectations of the interested parties (*stakeholders*). Project management has the aim of finishing a project inside schedule and within the defined budget, according to a previously arranged set of specifications. These elements characterize the Triple Constraint of project management, in which a project is made of three basic components: scope, time and cost. A well succeeded project, thus, means fitting these three objectives and satisfying the sponsors.

Internationally recognized by the effort on defining norms and supporting project management professionals, the Project Management Institute (PMI) published a general guide on project management: the PMBOK. The Project Management Book of Knowledge provides the best practices on project management that are applicable to the vast majority of the projects in many areas.

According to [5], the primary goal of the PMBOK Guide is to identify the subset of the Project Management Body of Knowledge that is generally recognized as good practice. But, in spite of being a well accepted guide, the PMBOK is not a process in the strict sense, as it does not define actions nor states how they must be followed and executed for the correct development of a project.

The PMBOK Guide does not include a metamodel. For this very reason, the managerial perspective of the integrated model proposed will use the metamodel designed by [6]. In order to compare and perform an integration of two models, they must be represented in compatible structures. As we can see in Fig. 1, that model covers concepts from general structures such as *Organization*, *Program* and *Project*, as well as the most important ones, such as *Activities*, *Stakeholders*, *Roles*, *Deliverables*, and associated classes. A fully detailed analysis of the classes on the metamodel is presented in section IV.

III. OPEN Metamodel

OPEN is an object oriented software development methodology maintained by the OPEN Consortium group [4]. It can be defined as a framework (OPEN Process Framework, OPF) which provides an extensible metamodel that can be configured for distinct software development processes. OPEN

encapsulates concepts and activities related to business, quality, analysis and reuse, that are common to any software development process using the object oriented approach.

A process is instantiated and customized from the OPEN metamodel by the addition and removal of process components [8], [9]. This operation helps in a better adequacy of the organization needs in terms of size, culture, investment and other characteristics, and involves choosing activities, tasks, techniques and specific configurations for the business.

OPEN's framework focuses on the cooperative interaction among the producers, their work units and what they produce [4]. Indeed, the OPEN Process framework (OPF) acknowledges the elements in Fig. 2 as the central components in its framework.

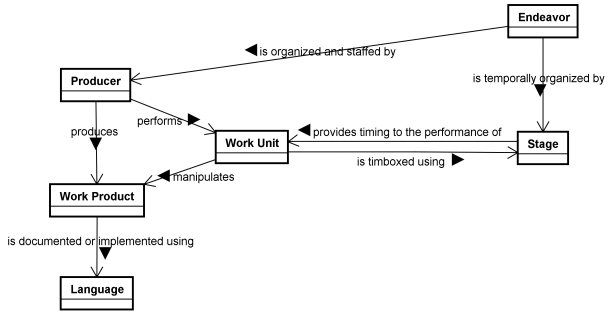

Fig. 2 Core Process Component Classes in OPEN [9]

The *Endeavor* class refers to a component that models the effort made from a *Producer* when executing *Work Units* during one or more *Stages*. *Work Product* refers to the components that are produced during the development of the project. The *Language* class models the type of language used to document and produce the project's products (e.g. UML, Java and even natural language). The *Producer* class refers to the element responsible for producing or modifying, either directly or indirectly, the artifacts of the process. *Producers* can be people (defined for *Roles*) or even tools. *Work units* consist of a set of cohesive operations performed by the producer to build a work, and they can be classified as Tasks, *Techniques*, *Workflows* and *Activities*. Finally, the time division dimension is provided by the *Stage* class, such as cycles, phases and instantaneous stages (like milestones).

IV. Comparative Analysis of PMBOK and OPEN Metamodels

In a previous study (see [6]) we have presented an integration of concepts arising from the PMBOK and RUP models. The detailed study of the PMBOK, RUP and OPEN metamodels helped to identify how their classes are organized and which are the valid relations between the elements of each model. The PMBOK metamodel includes the elements needed for project management while the concepts of software development processes are obtained by RUP and OPEN metamodels. The analysis of these software development processes metamodels (see Table I) allows us to identify elements of conformity between the central elements of RUP and OPEN. This comparative analysis is based on studies performed in [2] and [21] and add to the study presented by [22].

TABLE I
COMPARATIVE ANALYSIS BETWEEN RUP AND OPEN MAIN CONCEPTS

RUP	OPEN
Tool	Producer (Tool)
Tool Mentor	Work Unit (Technique)
Artifact	Work Product
Activity	Work Unit (Task)
Role	Producer (Role)
Discipline	Activity
Lifecycle	Stage (Lifecycle)
Phase	Stage(Phase)
Workflow Detail	Activity
Signature	-
-	Endeavor
-	Language

In RUP, the Tool class describes the tools that help the production or modification of an artifact. The OPEN metamodel also contains a class named Tool, which is a subclass of the Producer class, and represents a software used to create or modify versions of work products. In this case, we observed that the Tool class in RUP is equivalent to the Tool class in OPEN.

There is a similarity of concepts coming from the ToolMentor class in RUP and the Technique class in OPEN. The ToolMentor class is responsible for guidance on how activities are performed using a particular tool. The Technique class, a subclass of Work Unit class, is responsible for determining how to perform one or more activities, workflows and tasks by a producer.

According to the RUP, the Artifact class describes the types of work products that are produced and modified during the project. The Work Product class of OPEN models everything that is produced, used, modified or destroyed during the performance of one or more work units by one or more producers. In this case, there is a minimal difference between RUP and OPEN regarding the relationship of these two classes with a role (Role class). In RUP, an artifact should be the responsibility of only one role and can be modified by any or several roles. In OPEN, a work product must be related to one or more producers. Thus, the Artifact class of RUP is equivalent to the Work Product class of OPEN.

The definition of the Activity class in RUP meets the definition of the Task class in OPEN. The Activity class represents a work unit that produces a significant result for the project while the Task class models a specific work that produces or modifies one or more work products.

Both RUP and OPEN use the term Role to define who is responsible for performing the activities and produce or modify work products. In OPEN, the Role class is a subclass of Producer.

In RUP, the Discipline class is responsible for the division of the elements of the process in areas of interest. A discipline is composed of one or more workflows. Workflow Detail class groups related activities and defines how the roles should work

together to achieve specific objectives of the process. The latter class determines the sequence and interdependence between the activities belonging to a discipline. In OPEN, however, the set of elements based on a single activity, such as producers, work products and work units, which are part of a single field of knowledge is defined as the Activity class. Thus, in OPEN a discipline is organized around a single activity, which can contain multiple tasks. In addition, this class is composed of set of tasks, grouped according to a common goal, and produces a set of interconnected products where the dependency relationships between activities can be defined using pre-conditions and post-conditions. Consequently, the Activity class in OPEN encompasses the concepts of the Workflow Detail class and Discipline class in RUP.

According to RUP, the Lifecycle class defines the life cycle of software development. The OPEN proposes a division between product and process life cycles through the Business Engineering Cycle, Life Cycle e Development Cycle classes. Thus, the subclass Life Cycle class in OPEN is compatible with the Lifecycle class in RUP, as it represents the set of phases in which a single system, application, or main component is produced or used.

The Signature class in RUP contains two mutually exclusive attributes that indicate whether an attribute is used to input or output to a particular activity. In this case, it was not identified a similar class in the OPEN metamodel.

The Endeavor class in OPEN models the effort undertaken by producers during the execution of work units. This concept was not found in the RUP metamodel.

Finally, the Language class (which refers to the type of language used to document and produce the project), did not show compliance with any class of RUP.

V. Integrating PMBOK and OPEN models

The Meta Object Facility (MOF) provides a metadata management framework, and a set of metadata services to enable the development and interoperability of model and metadata driven systems [19]. This architecture model proposed by OMG is composed of four layers or levels. Then, a model defined on a higher layer defines the language to be used on the next lower layer [20]. When extending models it is important to be aware of the problems that may arise when representing concepts that belong to different levels of the MOF in a single diagram. In this paper, we are working with models that belong to the M1 layer (process model) of MOF.

The integrated model for the PMBOK and OPEN is composed of three packages: one for the project management concepts, one for the concepts of software processes (in this case OPEN), and finally, a common package that holds the concepts that occur in both models.

The OPEN metamodel classes represent the elements that compose process development software. The set of classes attributes defined in the OPEN package were based on [2], [8] and [9]. The PMBOK reference metamodel includes the required elements for project management. The main contribution of this model is the proposition of a set of classes and attributes (depicted in a UML class diagram) that corresponds to the concepts of general project management. The design of this model was based on [1] and [5].

When realizing an integration of two models, the conditions below may occur [6]:

- an *overlapping of concepts* (two classes with the same concept on each model) - in such case we can transform and join them into a single concept inside the common package;
- a *relation of concepts* (a class of one of the original models relates to some other class on the other original model, but they do not represent exactly the same concept), in such case we must create an association between them;
- classes with *independent and distinct concepts* from each original model, in such case we must leave each class in its own package.

In this model (see Fig. 3), the *Organization* class represents a company that is organized by programs (*Program* class). Programs are groups of projects (*Project* class) designated to reach a strategic objective. The organizations usually divide projects in several phases (*Phase* class) aiming a better managerial control. The definition of the project phases depends of the project type and adopted methodology but they are usually the following: Initiation, Development, Implementation and Closing. According to [1], a project should successfully finish each phase before beginning the next one. So, this way, there is a kind of dependence between the phases (relationship *PhaseDependency*).

Any necessary resource for the project, like people, equipment or place, is represented by *Resource* class. These resources are divided into active resources (*Stakeholder* class) and non-active (*PhysicalResource* class). Stakeholders correspond to all the individuals and organizations that have any kind of relation to a project [10]. The *OtherStakeholder* class represents a non-relevant stakeholder and the *RelevantStakeholder* class represents people and organizations whose interests are affected by the project. A resource can be either a direct member of the company's project team (*TeamMember* class) or a third party member of the company (*ThirdPartyMember* class). Also, the *PhysicalResource* class represents a physical resource in a project, such as a necessary material to accomplish an activity (*Material* class), a necessary equipment to accomplish an activity (Equipment class) or a physical place, for instance, a meeting room (Facility class).

The attribute *unitCost* contains the unitary cost of this resource and the attribute *timeUnit* contains the kind of time unit for this resource.

The *ProjectStakeholder* class was added to the model to represent the relationship of the stakeholder with the project. This class informs if it is a key stakeholder of the project (attribute *isKeyStakeHolder*), his level of interest in the project (attribute *levelOfInterest*) and his level of influence in

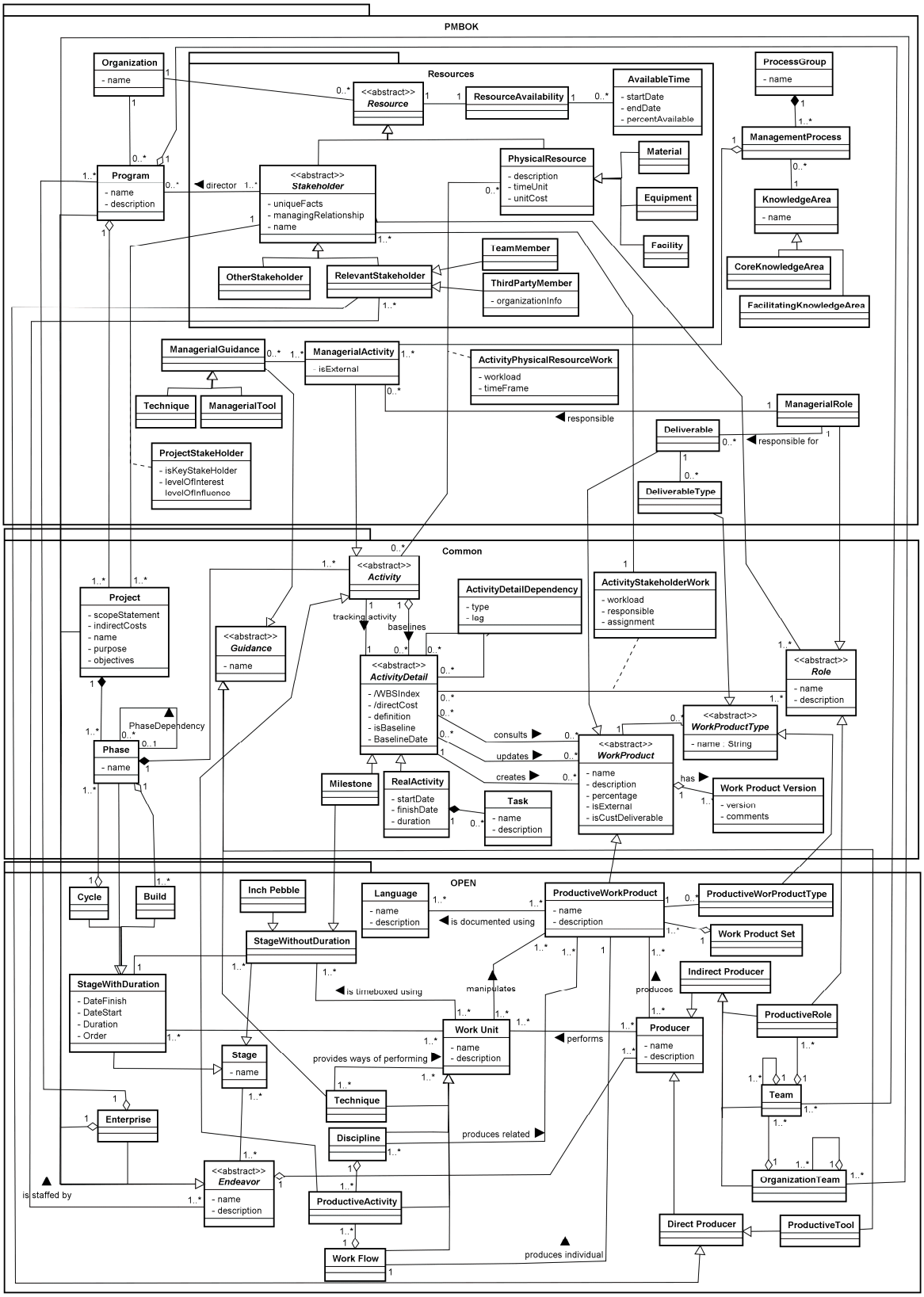

Fig. 3 PMBOK+OPEN Integrated Model

the project (attribute level of influence).The ResourceAvailability class adds relevant information regarding each resource's availability when assigning them to the activities, whether this is done manually or automatically. The *AvailableTime* class was added to the model to inform when a resource (such as a room or person) is available to be used. This availability is project independent, but it is not company independent. The attribute *percentAvailable* contains resource allocation percentage for a certain period. The attribute *startDate* and the atribute *endDate* defines the initial and final availability allocation date of a resource. In parallel, the *ActivityPhysicalResourceWork* class associates zero or more physical resources to zero or more activities (*Activity* class). It establishes the physical resources work load (attribute workload) in that activity. This relationship allows the automation of the resource allocation process, therefore an activity, for instance, can use a computer (resource) without the need of people's interaction.

The proposed model defines three different types of activities [11]. Activities directly related to the construction of the product, such as coding or database modeling, are called productive activities (*ProductiveActivity* class). Managerial activities (*ManagerialActivity* class), however, may belong to the software development workflow (attribute *isExternal* = false) or belong to the business organization workflow (attribute *isExternal* = true). Activities that are only necessary to coordinate the construction of the product are referred to as managerial activities. Any other activities that do not belong to an individual project's activity workflow (and may be else shared by other projects) are called management supporting activities. Following this nomenclature, the activity of organizing and conducting a follow-up meeting of the project is an example of a managerial activity that belongs exclusively to the software development project. In contrast, the activity of hiring a database administrator is an example of a management supporting activity that belongs exclusively to the other enterprise workflows to support the project activities of the organization (in this case, this activity is performed by the human resources department).

Each activity can belong to one or more baselines. In each baseline generation, an activity should maintain the relationships with the roles and work products (*WorkProduct* class). Thus, the *ActivityDetail* class was defined as responsible for maintaining these relationships, while the *Activity* class was defined as an aggregation of one or more *ActivityDetail* classes. This class will be responsible for storing pertinent information about the execution of an activity and, mainly, for maintaining the relationship between activities. Each activity should have a defined work product, a given responsible role, involved resources (human and material), the necessary effort to execute the activity, the dependence relationships with other activities, the necessary time (duration) to execute the activity, and, cost information. The field *isBaseline* represents a specific activity of the baseline.

The *ActivityDetailDependency* class defines the sequence of activities in a project. This class also defines if one or more activities can be executed in parallel, and if two activities can be overlapped. In addition, an activity can have a duration of time to be defined (*RealActivity* class), has a starting date and a finishing date, and may be subdivided in tasks (*Task* class) or an activity may not have a duration of time (*Milestone* class). Thus, while the relationship called *baselines* allows an activity to belong to one or more baselines, the relationship *trackingActivity* differentiates the current activity of those that belong to the baselines.

Stakeholders can play several roles (*Role* class) during the execution of project activities. Thus, for each association between a role and activity (*ActivityStakeholderWork* class) there must be an association of this activity with a stakeholder able to play that role. Moreover, as the concept of roles appears in both models (PMBOK and OPEN), those were divided into managerial roles (*ManagerialRole* class) and productive roles (*ProductiveRole* class). Then, managerial activities are performed by managerial roles and productive activities are performed by productive roles. In addition, the *Team* class defines a collection of one or more related roles which collaborate to perform productive activities. So, one role can take part of multiple teams and a team can contain multiple roles. Also, the *OrganizationTeam* class is originally called Organization in OPEN's metamodel and consists of a cohesive collection of teams. It is responsible for dividing the company's human resources into smaller and more manageable organizational units.

The *WorkProduct* class represents something that is produced, consumed or modified (such as documents, models or source codes) during the execution of activities. A work product should be associated to one role, which is formally responsible for the production of this work product. The attribute *isExternal* indicates that the work product should be approved by the sponsor or the customer. The attribute percentage contains the development rate of the product. Also, a work product can be subdivided in managerial products (*Deliverable* class) or productive products (*ProdutiveWorkProduct* class). The *WorkProductType* abstract class contains information about the type of a work product in a specific software project. Thus, the *DeliverableType* class describes a category of managerial work product, such as meeting minutes, and the *ProdutiveWorkProduct* describes a category of productive work product, such as UML model or code library. The *ProductiveWorkProduct* class is originally called *WorkProduct* in OPEN's metamodel. It is a core method component that represents a work product that is produced, consumed or modified during the execution of productive activities by productive roles.

The cohesive set of products produced by one or more tasks of activities is represented by the *WorkProductSet* class, while the *WorkProductVersion* class corresponds to a specific version of the product. The *Language* class models the languages used to document work products.

The PMBOK Guide represents their practices in two logical dimensions. One dimension defines nine knowledge areas (*KnowledgeArea* class) while the other dimension describes thirty-nine management processes of a project (*ManagementProcess* class) that are organized in five process groups (*ProcessGroup* class). The knowledge areas are classified as core knowledge area (*CoreKnowledgeArea* class), such as scope, team, cost and quality, or facilitating knowledge area (*FacilitatingKnowledgeArea* class), such as human resource, communications, risk and procurement. Therefore, each managerial activity belongs to a management process and is also related to a knowledge area.

The *Endeavor* class describes a core method component that models an effort undertaken by collaborating producers during multiple stages to develop and maintain related applications. The OPEN metamodel defines the following endeavor's subclasses: *Enterprise*, *Program* and *Project*. The *Enterprise* class represents the highest level endeavor, consisting of a collection of related programs which are managed as a single unit.

The *Producer* class describes a core method component that provides related services and produces, either directly or indirectly, versions of related work products. It is subdivided in direct producers (people and tools) and indirect producers (organization, team and role). Also, producers should fulfill their responsibilities by performing their tasks and collaborating with other producers.

Work units (*WorkUnit* class) are method components that model functionally cohesive operations that are performed by producers during the delivery process. These are classified as disciplines, techniques, workflows and activities. The *Discipline* class models a collection of productive activities that has a common objective. A discipline produces a set of one or more related work products. The *Workflow* class consists of a collection of productive activities that either produces a single work product or provides a single service. The *Technique* class is responsible for modeling a way of executing one or more work units. Finally, the *ProductiveActivity* class describes the productive activities accomplished by productive roles.

The *Stage* class represents a core method component that models time intervals that provide a macro organization to the work units. This class is subdivided in stages with duration (*StageWithDuration* class), such as cycles, phases and builds, and stages without duration (*StageWithoutDuration* class), such as milestones and inch-pebbles. The Cycle class represents a period of time when one or more work units can be executed. A cycle consists of one or more phases. The *Build* class is responsible for decomposing the phases in manageable periods of time. These periods of time should have a short duration (such as, one day or one month). The *InchPebble* class represents miniature milestones.

The *Guidance* class represents the process orientation elements. It describes the use of techniques (*Technique* class) and necessary tools (*ManagerialTool* and *ProductiveTool* classes) for the execution of some activities. Techniques help to define the required skills to perform specific types of activities.

An earlier work [6] developed an integration metamodel between PMBOK and RUP classes (called PMBOK+RUP). The integration of these concepts produced a set of 19 rules (see [11]) to ensure the consistency of the model. This study allowed the development of a methodology for integrating models of project management with models for software development processes. As a result, the integration metamodel between PMBOK and OPEN (called PMBOK+OPEN) has a similar structure to the PMBOK+RUP model (replacing the package for the software development process). The two software development processes, however, have particular characteristics that are reflected in different classes and different relationships with the PMBOK and Common packages.

Based on the results of current research, 10 new rules (see Table II) were added to the 19 rules developed previously in [11]. These constraints could not be expressed in the diagram due to limitations in the expressiveness of the UML class diagram.

TABLE II
ADDITIONAL CONSTRAINTS ON INTEGRATED MODEL

#	Constraints
1.	A discipline models only a collection of productive activities that has a common objective;
2.	Products can be documented using several languages. But a specific product should be documented only with a specific language;
3.	A managerial tool cannot be related to activities that produce or modify a productive work product, only a managerial work product. However, it can consult a productive work product;
4.	A productive tool cannot be related to activities that produce or modify a managerial work product, only a productive work product. However, it can consult a managerial work product;
5.	A organizational team is cohesive collection of teams. Then, an organizational team cannot contain teams that have only managerial roles. Thus, there would be no productive work products;
6.	A organizational team is cohesive collection of teams. Then, an organizational team cannot contain teams that have only productive roles. Thus, there would be no managerial work products;
7.	Productive work products must be documented with a so called productive language;
8.	Managerial work products must be documented with a so called managerial language;
9.	The duration of the phases is calculated by adding the time of their associated activities;
10.	The cycles of a software project cannot proceed in parallel.

The PMBOK+RUP and PMBOK+OPEN metamodels provided the conceptual framework necessary to develop a unique model to assist in project planning considering the concepts arising from the software development processes. To demonstrate the feasibility of proposed concepts, we developed an integrated model called SPIM - Software Planning Integrated Model (see [11]). Based on this idea, the concepts coming from the integration between the PMBOK and OPEN were added to the SPIM integrated model (which included only the integration of the PMBOK and RUP). In

order to illustrate the practicability of the concepts proposed by the SPIM model and its set of rules, we developed a prototype called Software Planning Integrated Tool (SPIT). The SPIT (Fig.4) was developed in C# and acts as an add-in for Microsoft Office Project 2007. This choice allows SPIT to take advantage of the features that are already implemented in accordance with the proposed integration model in this software for project management. All information needed to perform the validations of SPIM is stored in custom fields inside the commercial software.

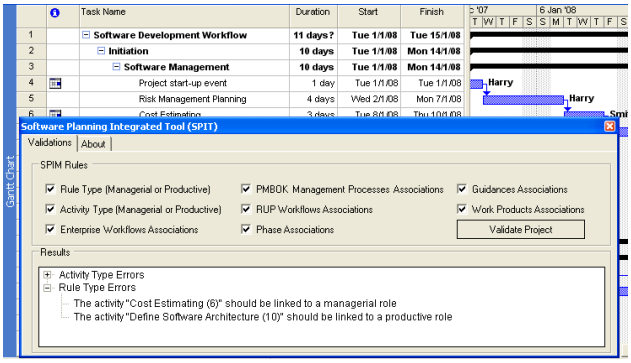

Fig. 4 SPIT in action

VI. Conclusive Remarks and Future Work

This paper presented a proposal for the integration of PMBOK's main concepts with a model for software development process, namely, OPEN. We first have identified the importance of project management activities during a software development project. Then we noticed the lack of information about these skills in most software development processes today. After an individual analysis of each base model, we proposed a new metamodel that covers both perspectives into a single integrated model.

According to many empirical studies, the effectiveness of an organization depends, in some part, on the success of its projects [10], [12]. Still many researchers work on the investigation of projects' success factors, such as product definition, execution quality, and project management techniques [10], [13].

In a simplified view, a software development process is a set of activities and related results that lead to the production of a software. According to [14], a software development process is the set of the necessary activities to transform user requirements in software products. The importance of having a standard software development process relies on the fact that it becomes the guide for the execution of all projects inside an organization. Thus, many processes such as RUP, Extreme Programming (XP) [15], Microsoft Solutions Framework (MSF) [16] and OPEN are being used as a common ground when designing the standard software development processes.

According to [17], a software development process is one of the main responsible mechanisms to manage and control projects and software products. Thus, applying project management knowledge along with an appropriate software development process makes it possible to obtain a more complete flow of project management and software development.

We must remark that the proposed model here is an evolution of the PMBOK+RUP integrated model presented in [6]. So, it follows the objectives below:

- it should allow the integrated planning of the product and management of the project;
- it should make the distinction of activity types (managerial or productive) and work products;
- it should allow the integrated scheduling of managerial and productive activities;
- it should add the notion of availability of a resource, so that this information can be used to automate the resource allocation processes in software projects;
- it should add the notion of availability to a resource. It can be used to automate the resource allocation in projects of software development;
- it should preview workload information for the associations of a role, activity and stakeholder;
- it should distinguish the possible relations between an activity and an artifact (create/update/consult);
- it should also allow the integration of project management concepts provided in PMBOK with the concepts of software development provided in OPEN.

This work brings new interesting finds which reaffirm the goal of designing a support tool for software project managers. During the development of this research we have identified the need for more development on the items below:

- Definition of constraints for the integrated metamodel via OCL (Object Constraint Language);
- Extending this integration metamodel to other software development processes, such as XP and MSF;
- Evaluation of the proposed model with software companies, using the prototype in real projects.

We believe that is possible to extend this integration metamodel to other software development processes because, in agreement with [18], different models of software development processes share fundamental activities, such as: software specification, project and software implementation, software validation and software evolution.

By performing this integration with different software processes and models we can bring new interesting questions and provide a more comprehensive approach to the software engineering field. For each one of these studies researchers might mainly depart from the concepts found in the PMBOK, a well-accepted document that contains the so-called best practices for the vast majority of projects in many areas, including software development.

This research adds to the field by proposing the same approach with the OPEN Process Framework in order to bring new interesting approaches and discussions. The next steps of this research indicate new contributions for the software engineering area, improving our understanding of project management's relationships to software development projects.

REFERENCES

[1] K. Schwalbe, “Information Technology Project Management”, Thomson Learning, 2nd edition, 2002.
[2] B. Henderson-Sellers, R. Due, I. Graham, and G. Collins, “Third generation OO processes: a critique of RUP and OPEN from a project management perspective”, Seventh Asia-Pacific Software Engineering Conference, 2000.
[3] P. Kruchten, “The Rational Unified Process: An Introduction”, Addison-Wesley, 2nd edition, 2000.
[4] I. Graham, B. Henderson-Sellers and H. Younessi, “The OPEN Process Specification”, Addison-Wesley, 1997.
[5] Project Management Institute, “PMBOK - A Guide to the Project Management Body of Knowledge” Newtown Square, PA: Project Management Institute, 4th edition, 2008.
[6] D. Callegari, and R. Bastos, “Project Management and Software Development Processes: Integrating RUP and PMBOK”, ICSEM - International Conference on Systems Engineering and Modeling, 2007.
[7] R. Pressman, “Software engineering: a practitioner's approach”, McGraw-Hill, 6th edition, 2004.
[8] OPEN Consortium. “OPEN - Object-oriented Process, Environment and Notation”, viewed March 05, 2011, < http://www.open.org.au>.
[9] OPEN Process Framework Repository Organization (OPFRO), “OPEN Process Framework”, viewed February 15, 2011,<http://www.opfro.org/index.html>.
[10] R.G. Cooper, “Winning at New Products: Accelerating the Process from Idea to Launch”, Perseus Books, 3rd edition, 2001.
[11] D. Callegari, M. Rosito, M. Blois, R. Bastos. “An Integrated Model for Managerial and Productive Activities in Software Development”. In: ICEIS - 10th International Conference on Enterprise Information Systems, Spain, 8p, 2008.
[12] H. Kerzner, “Applied project management: best practices on implementation”, New York, Wiley & Sons, 2000.
[13] J. Pinto, and D. Slevin, “Critical factors in successful project implementation”, IEEE Trans Eng Manage, 34(1): 22–7, 1987.
[14] I. Jacobson, G. Booch, and J. Rumbaugh, “The Unified Software Development Process”, Upper Saddle River, Addison Wesley, 2001.
[15] K. Beck, “Extreme Programming Explained: Embrace Change”, Addison Wesley, 2nd edition, 2004.
[16] M. Turner, “Microsoft Solutions Framework Essentials: Building Successful Technology Solutions”, Microsoft Press, 2006.
[17] W. S. Humphrey, T. R. Snyder, and R. R. Willis, “Software Process Improvement at Hughes Aircraft”, In: Institute of Electrical and Electronic Engineers - IEEE, p. 11-23, 1991.
[18] I. Sommerville, “Software engineering”, Addison-Wesley, 8th edition, 2006.
[19] OMG, “OMG Meta Object Facility (MOF) Core Specification”, viewed February 20, 2011, <http://www.omg.org/spec/MOF/2.4.1/PDF>.
[20] OMG, “OMG Meta Object Facility (MOF) Core Specification”, viewed February 21, 2011, <http://www.omg.org/spec/SPEM/2.0/PDF>.
[21] OMG, “Software & Systems Process Engineering Meta-Model Specification”, Version 2.0, viewed January 15, 2011, <http://www.omg.org/spec/SPEM/2.0/PDF/>.
[22] M. Rosito, D. Callegari, R. Bastos, “Metamodelos de processos de desenvolvimento de software: Um estudo comparativo”, SBSI - Brazilian Symposium on Information Systems, 2006.