Object Management Group
Framingham Corporate Center
492 Old Connecticut Path
Framingham, MA 01701-4568
U.S.A.

Telephone: +1-508-820 4300
Facsimile: +1-508-820 4303

Analysis and Design Platform Task Force

Process Working Group White Paper on Analysis & Design Process Engineering

Version 1.0

OMG Document ad/98-07-12

29 July 1998

Table of Contents

Appendix B - Example SPDL Meta-Model

1. Introduction

1.1 Analysis and Design Task Force Process Working Group Mission

The Process Working Group of the Analysis and Design Task Force (ADTF) is to provide a forum for identifying and building consensus and convergence in the industry around system development process/methodology in a distributed environment. In addition, it will develop and manage process white papers, RFIs and RFPs for the ADTF.[1]

The first formal effort of this group is this white paper to address software and software system process engineering.

1.2 Industry Problem

1.2.1 State of the Current Practice

Process engineering in the software development industry is not a universal well defined and standardized activity. It has a high degree of variability due to a large number of factors. It varies according to cultural aspects, practices, and habits. It depends on the type of project (size, critical aspects, novelty, repetitiveness, etc.), on the size of the organization, on the standards that are used, on the technology that is used, and on the domain for which software is developed. It also depends on the end user requirements regarding the software, and in particular the quality criteria.

In addition, the UML 1.1 has been adopted by the OMG as a modeling language and has received wide recognition and acceptance in the industry. The UML addresses modeling with associated diagrams, notation, semantics, etc. It does not address process., nor was that its intention.[2]

Analysts, designers and developers need assistance in how to use the UML and other software development work products.

1.2.2 Maturity Level of the Industry

Institutionalization is a major step for the process maturity in an organization.[3] It entails building an infrastructure and a corporate culture that supports the methods, practices, and procedures of the business so that they last after those who originally defined them have gone. As a software organization gains in software process maturity, it institutionalizes its software process via policies, standards, and organizational structures. Defining, improving and institutionalizing processes are the main activities for " process engineering ".

In practice however, many companies have invested much time and money in process, but the results have not be satisfactory in many cases due to the constant technical, market, and

[1] ADTF Process Working Group Mission Statement ad/98-03-02
[2] Process was specifically excluded from the RFP in response to the feedback received in the original RFI which said that there was not and may never be convergence in the industry on process.
[3] *Capability Maturity Model, Version 1.1,* Software Engineering Institute

standards changes; culture resistance; and lack of sustained management support. Very often, the process definition work is reinvented for every project in every company. The mergers of companies, or of development teams, or the inter-project cooperation, unavoidably leads to a new study of the procedures.

1.3 Purpose of the White Paper

1.3.1 General Purpose

The purpose of this paper is to describe a means for software development process engineering to address the issues of:

- how to effectively use the UML in specific software/software development projects, and
- the maturity of the development of systems and software.

This paper is intended to describe and obtain agreement on the concept that processes and methods for software development can and should be described using a ***software development process definition language***. This language would be the foundation for engineering project specific software development processes and methods. This engineering, in essence a "methodology to create methodologies", would permit individual software development projects to select the components for their processes and methods based on that project's specific goals, risks, capabilities, etc.

Further, this paper will provide the foundation for possible RFIs and/or RFPs for the process definition language and models.

1.3.2 No Single Process

This paper does not propose that there is a single process and/or methodology. Rather, it asserts that it is not possible to standardize a process or methodology for all software development situations. For example, to quote from UML 1.0:

> *"Processes must be tailored to the organization, culture and problem domain at hand. What works in one context would be a disaster in another. The selection of a particular process will vary greatly, depending on the things like problem domain, implementation technology, and skills of the team."*

Further, it is probably not possible even to standardize on the names of the software development phases. For example, there has been an ongoing discussion in the OTUG mailing list on whether there is even is such a thing as an Object-Oriented Analysis Phase. However, this inability to agree on even some of the basics, need not stop progress. There are some basic things in common to all software developments - objectives/goals, risks, activities, work products, etc.

1.3.3 Single Process Definition Language

While it may not be possible to make much progress on standardizing the processes and methods, we assert that it is possible to standardize the language to specify them. There are over 50 existing published object-oriented methodologies. These methodologies can be restated (with effort of course) in that language. This standard language would then allow projects and organizations to compare and contrast different methodologies properly and engineer the processes and methods based on their own specific circumstances.

Indeed, if the language was standardized, the tool market could eventually automate some of software process engineering. Computer Assisted Process Engineering (CAPE) and Computer Assisted Software Engineering (CASE) tools exist today. This effort will provide the conceptual foundations to enable the integration and/or inter-operation of these two types of tools. The underneath CORBA technology would then make it possible to share, distribute, exchange and extend process and method elements.

Finally, even though the concepts described in this paper ***may*** apply to process engineering for anything, it is not the intent of this paper to describe general process engineering. The focus is on resolving the issues of effective UML usage and process maturity in software development projects.

1.4 Intended Audience(s)

This white paper is intended for the OMG as a whole and, specifically, those task forces and working groups which are working in the areas of workflow and processes. This will ensure the work in these workspaces is well synchronized.

Another audience may be the process and methodology industries outside of the OMG. Their thoughts and suggestions will help address the commercial significance of the concepts and directions established in the paper.

1.5 Writing Team

Editor: Mike Bradley, BellSouth.

Contributors: Philippe Desfray, Softeam
Michael Jesse Chonoles, Lockheed Martin Advanced Concepts Center
Paul Allen, Select Software
Steve Tockey, Rockwell Collins
Van-Si Nguyen, Xerox

1.6 Questions & Comments

Please address content-related questions or feedback to the Process Working Group and to the paper editor Mike Bradley (mike.j.bradley@bridge.bellsouth.com). In the interests of mailing list efficiency, please address errata or omissions directly to the editor only.

2. Definition of Process Engineering

A process is a system of operations for producing something, in other words, a series of actions, changes or functions, that reach an end or a result. A software process is a set of activities, methods, practices, and transformations that people use to develop, maintain, acquire, and subcontract software and the associated products (documentation, code, etc.).[4]

Process Engineering (PE) covers the activities related to the definition, optimization, and control of processes. For software development, the processes to be engineered are those used in developing software.

3. The Scope of Process Engineering Addressed in This Paper

This paper covers process engineering for defining the processes to be used in *software* development. This scope could also be interpreted as process engineering for *systems* development in which the effort includes the development of software.

Thus, Software Process Engineering (SPE) is, in effect, the selection of the specific software development activities, the order in which they will be carried out, etc. The activity of software process engineering is considered to be one of the components of the planning for software development projects. The positions expressed in this white paper are related to those processes whose results are likely to be expressed in UML.

Even though the concepts described in this paper ***may*** apply to process engineering for anything, it is not the intent of this paper to describe process engineering in general.

4. Software Development Process Definition Language and Meta-Model

A ***software development process definition language*** is needed as a foundation for the process engineering of software development processes and methods for individual projects. Further, this language needs to be based on software engineering components using a welldefined meta-model.

4.1 Software Development Process Definition Language

The goal of the **Software Process Definition Language** (SPDL) is ***not*** to define a language resembling a syntactic or graphical language such as C++ or UML, but rather to define and standardize a formal structure for specifying any particular software process.

The SPDL will allow different software processes to be engineered appropriate to different project profiles. A project profile will reflect such things as the organizational culture (hierarchical vs. empowered teams) , industry sector or domain (insurance, telecommunications etc.), technology type and combinations (relational database, Internet, C++ mainframe, VB etc.), etc.

[4] *Capability Maturity Model for Software, Version 1.1,* Software Engineering Institute

The SPDL will be used to describe existing processes and methodologies so that projects can engineer their projects potentially using parts of many processes or methodologies according to the project's profile. In addition, it will make it possible to:

- introduce new types of process engineering components,
- define its relationships to other components,
- combine components,
- customize existing components, and
- have these components shared between different projects and organizations.

Also, the structure will allow the software community to share, distribute, reuse, automate, and customize the defined processes. The language will address such elements as software development work products, activities, tasks, techniques, methods, etc. In addition it will address project profile information such as goals, risks, organization, resources and their skills, technologies to be used, roles, metrics, rules, etc.

4.2 Well Defined Meta-Model

In the OMG context, underlying the SPDL will be a meta-model described using:

- the MOF (Meta Object Facility) for meta-model interoperability and CORBA interface generation
- the UML for notation.

This UML meta-model is not a system development project deliverable, but rather it describes the possible process components from which each project will select to develop its own process and methods.

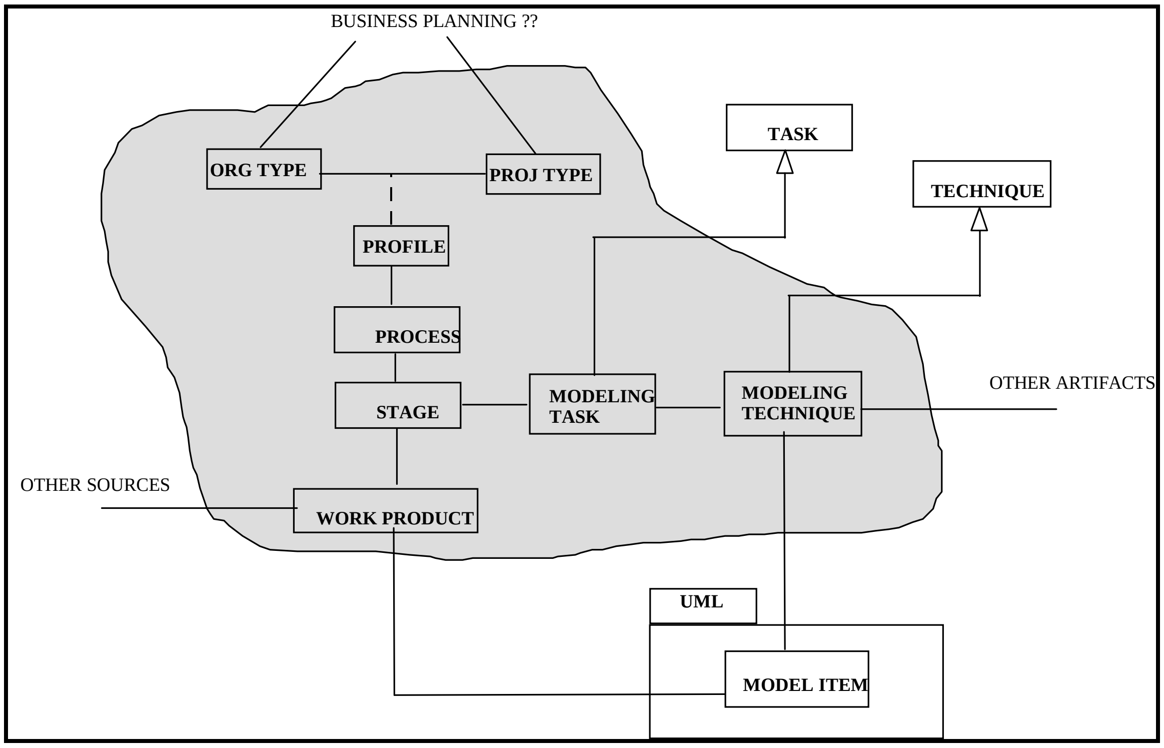


Figure 1: Example Meta-model and Software Process Engineer's View .

Figure 1 is an example of a possible meta-model for SPDL. The meta-model would contain such concepts as Organization and its capabilities; Project and its characteristics; tasks; etc. In addition, as this example shows, the meta-model would point to other already defined meta-models such as UML1.1. This will allow the definition of software processes for specific projects to be supported by tools.

Thus, for an individual project, the software process engineering results in a unique software development process (described in the language of the SPDL meta-model) that describes which of the UML models will be produced in the project, as well as, how they will be produced, in what level of detail at each phase, how they will relate to other models, etc. For example, the engineered process might specify a particular UML diagram with little formal rigor to provide a discussion vehicle at system proposal or feasibility stage. Further, at the other extreme it might specify the same UML diagram with a high degree of rigor later in the project life-cycle for the specification of the delivered software.

5. Process Engineering

SPE is used by process engineers in project planning and by practitioners in executing the project. Practitioners are software developers, project leaders, and other roles on a software project which are likely to be involved in modeling - significantly this includes users.

5.1 Software Process Engineers

Figure 1 in the preceding section, is an example of how the SPDL could be scoped in the context of the process engineer's domain of interest.

The flexibility of the language is of profound importance here. Note the de-coupling of the UML model, which has its own syntax, semantics and rules of consistency. Techniques are further de-coupled from tasks. A technique is a procedure which guides the creation of one or more UML model items. A technique is modular in that it may be used by different tasks. A task is in turn also modular in that it can be used within different stages of different processes. This provides great flexibility while at the same time retaining the soundness and stability of the underlying UML model.[5]

So, what's the bottom-line? Where's the value?

The process engineer can engineer the process and methods for a software development project using the SPDL (which is based upon the foundation of a well defined meta-model) using the project's profile. This will result in techniques, tasks, deliverables and stages that make up the processes and methods of relevance to his/her organization's project. The process engineer can then give guidance to the project as illustrated in Table 1.

5.2 Software Development Practitioners

[5] Note: there are two other more detailed examples in Appendix A and B.

Table 1 represents some of the types of questions practitioners might ask and how the process and methods developed by the process engineer for the specific project might answer them.

Question	Example response
what is the overall shape and sequencing of the process?	spiral, waterfall, …. suitable diagram to show structure and sequencing.
what deliverables are expected, and when?	project proposal after inception stage.
which models are appropriate (and why) ?	class model - to capture system structure.
which models are mandatory/optional parts of deliverables produced at each stages of development?	class model is mandatory within analysis doc after analysis stage.
what level of model detail is needed for each deliverable?	class model must include attribute/operation descriptions within analysis doc.
what are the required inter-relationships between models at each deliverable?	a use case must correspond to either a sequence diagram or a collaboration diagram in the external design specification.
what rules of thumb apply to a model at each deliverable?	a service package contains around 5 to 15 classes in the external design specification.
who (team role) is responsible for those models?	system architect is responsible for service package dependency diagram.
which modeling techniques are appropriate?	use case, class, collaboration…….
how is a technique performed, what are the procedural guidelines?	identifying classes is performed like this……
what rules of thumb (heuristics) apply when using a technique?	in identifying actors look for the roles (salesperson) not the individuals (Fred, Mary, Joe..)
when should/must the techniques be used, what is the recommended sequence of techniques?	use cases must be identified before modeling collaborations.
which modeling techniques are mandatory/optional at different stages of development?	class modeling is mandatory in analysis.

TABLE 1: PRACTIONER'S DOMAIN OF INTEREST.

6. Considerations for A&D Process RFI/RFPs

It is anticipated that RFIs and/or RFPs will be issued for a meta-models and SPDL. Section 5, Appendix A and Appendix B contain some partial examples of possible languages and meta-models using different approaches.

6.1 Planned Roadmap

- RFI to validate the contents of this Whitepaper.
- RFP for the Meta-Model and Software Process Definition Language (SPDL)
- RFI or RFP for processes to use
- RFPs to address other needs identified in the original RFI.

7. Other Similar Standards and Efforts

While it would appear that many "process standards" efforts such as SEI/CMM, ISO/IEC 15504, BOOTSTRAP, ISO 9000, ISO/IEC 12207, MIL-STD 498, Trillium, The V-Model, these efforts do not directly address an SPDL supported by a well defined meta-model. These standards would be potential processes to be described by an SPDL or they could help define and/or constrain other processes described by an SPDL.

8. Relationship to OMG Technology and Other Work Efforts

8.1 Workflow RFP

The feeling is that this effort is complementary to the BODTF (CF) RFP-2, *cf/97-05-06 Workflow Management Facility*. While the target of this white paper is processes for software development, any technology adopted as a consequence of it may provide process definitions to be used by the Workflow Management Facility. The main interface between the workflow RFP and any technology resulting from this white paper would be interfaces to the process definition repository where the process "instances" would be accessible to the workflow management facility.

In addition the workflow tends to focus more on the execution and control of processes, including nested processes. The workflow processes may change during their execution, but, in general, they are not individually created for each lot, run, etc. On the other hand, SPDL tends to focus more on the creation of a process from a repository on a project-by-project basis depending on the goals, risks, capabilities, and other characteristics of the specific project and organization doing the development.

8.2 UML

Figure 2 shows the three main relationships between the SPDL meta-model and the UML meta-model. These links must be carefully studied in order to integrate the two meta-models and other areas of process engineering and UML notation and diagrams properly. Responses to any Software Process Engineering RFPs must clarify such links.

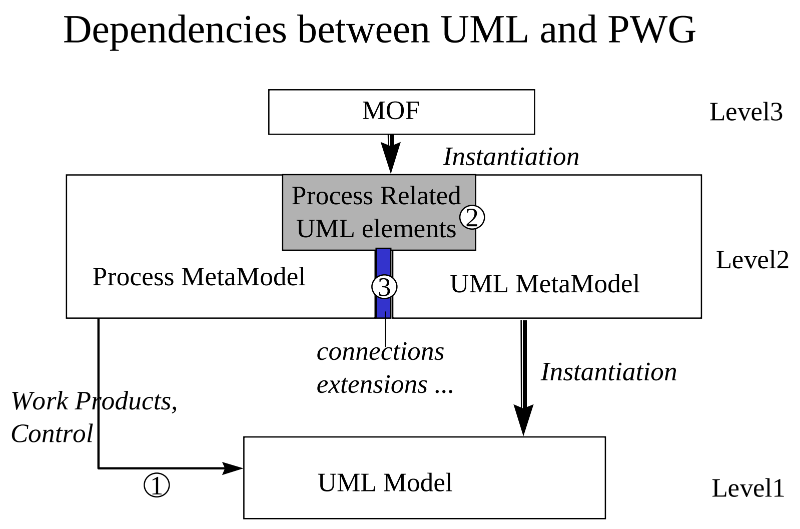


Figure 2: Connections between SPDL and UML Meta-Models

8.2.1 SPDL and UML Model Elements

SPDL and UML Model Elements are related as represented by Link 1 in Figure 2. The effective use of the UML is the main focus of the process engineering working group. It is expected that many UML elements such as diagrams, components, packages, etc. will be related to process activities using an SPDL. An SPDL can be used toidentify which UML elements will be work products in the process.

8.2.2 UML Model Elements Related to Process Engineering

Another aspect to be dealt with is the management of UML elements related to process engineering. This connection is represented by Link 2 in Figure 2. For example, stereotypes and tagged values may be restricted to only certain activities, or life-cycle phases. These restrictions must be expressible by any SPDL. For example, during analysis, certain tagged values must be used while others are dedicated to design. Tagged values and stereotypes restrictions can exist at every part of the life-cycle. These UML notions have different meanings and are not necessarily available, depending on the stage of the life-cycle.

In addition, there are relationships among such things as UML diagrams, techniques, and activities which must be represented in the process meta-model. In other words, how do the UML “Model Elements” (in the UML 1.1 meta-model sense) relate to “process elements” that are likely to be defined in a process meta-model. As developed in Appendix A, some mapping techniques such as the point-of-view notation can be made.

8.2.3 Meta-Model Alignment

The meta-model alignment is represented by the Link 3 in Figure 2. There are alignment constraints between the UML meta-model, and the process engineering meta-model. As an example, the activity diagram meta-model may be related to the process engineering notion of activity.

There may even be the need for extensions of the UML meta-model. For example, process definition may introduce new requirements on the content and presentation of UML diagrams. This would in turn introduce new requirements to the UML meta-model.

8.3 Manufacturing Task Force

The SPDL and manufacturing processes, tasks, resources, organizations, etc., while they are the same at some high level of abstraction, are not similar in the way they are used. In manufacturing the focus, like in workflow, tends to be more on the execution and control of processes which are not individually created for each lot, run, etc. On the other hand, SPDL tends to focus more on the creation of a process from a repository on a project-by-project basis depending on the characteristics of the specific project and the organization doing the development.

8.4 ORMSC/RMWG - Viewpoints (Van-si)

Appendix A - Characteristics of an SPDL

9. Goal of the language

There is a tremendous need to formalize process engineering components, to reuse those that are already defined, to have them shared between organizations with common interests, and to go on improving them. This can only be obtained if a framework is defined and standardized.

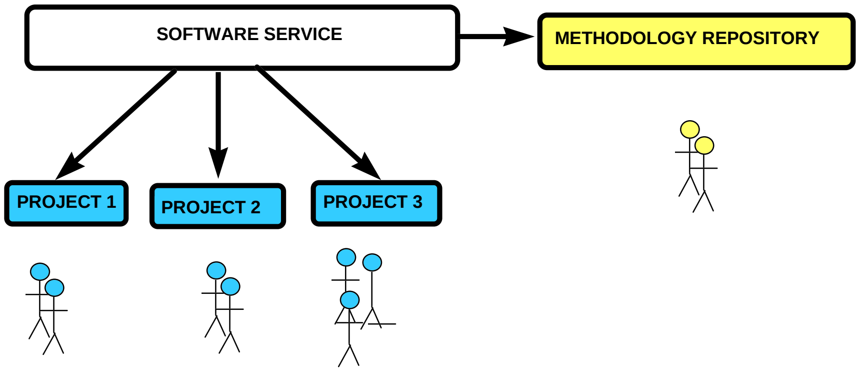


Figure A-1 - *Distributing the methodological skill to several projects*

This Framework should make it possible to introduce every new kind of process engineering component, to define its relationships to other components, to combine components, to customize existing components, and to have these components shared between different organisms. Process engineering components such as “ activity ”, “ rule ”, “ document template ”, “ work product ” need to be supported by the framework.

9.1 Introduction

As opposed to software that is made of *work products*, process engineering is made of everything that helps to produce the *work products*. Process engineering deals with such things as:

- *development technologies* used for software,
- *system development activities* needed for developing a software,
- *rules or guidelines* that must be followed by the developer,
- risk management for the work products elaboration,
- goal definition,
- project organization,
- roles,
- metrics, and
- skill management.

Note: This list is not meant to be exhaustive.

There is an orthogonal relationship between work products or development technologies, and process engineering. For example, many different methodologies can be used with the UML technology, or any other technology such as idl, C++, Java, etc.

In addition, process engineering deals with “ combining technologies ”. For example, it can define how to combine the UML and C++ technologies during a detailed design. This mapping needs to be supported by the process definition language.

We may not be exhaustive in describing every type of process engineering element, but we may formalize the main supported and distributed elements that can be identified, and provide an extensibility mechanism for that purpose.

9.2 Work products

Work Products are the fundamental material that process engineering has to manage. Work products define each element that has to be produced in an automated or manual way, using a set of technologies. These can be for example UML models, documentation, reports, code source, binaries, etc. Metrics can also, but not entirely, be defined as work products.

Process engineering specifies every work product that has to be produced during a software development. It also defines every activity that will produce all these work products.

Further, for every kind of work product, it will provide dedicated rules and guidelines. It states exactly how a work product should be produced, what elements have to be inside. It also defines how the work product will be controlled, how it will be identified, maintained, what organization will manage it, what is its goal, what risks are related to its development, etc.

Finally, process engineering specifies the situations for when the work product is to be used or not used. These goals and risk avoidance criteria are used to match the work products to specific projects.

9.3 System development activities

An activity is the process unit that will produce one or more work products. An activity needs resources that can be specified in terms of the roles, as well as input work products, it has links with other activities, and is related to dedicated procedures (quality control, termination criteria, metrics, etc.).

The activity elements constitute the main connection between process engineering, and project management. Project management has not been recognized as being a main target for process engineering (SLC meeting decision). For that reason, activities may optionally provide details on schedule information.

9.4 Organization, roles

Roles must be defined in order to clearly specify the responsibilities each should have in producing a product in an activity. The project team organization has to be expressed through the roles of every resource involved. Project manager, developer, architect, tester, configuration manager, are examples of roles. Each of them is specialized in some activities, and has specific duties and responsibilities regarding specific work products.

Tools strongly influence organization, for they automate or help the developer’s tasks. The work they provide has to be managed as well. A compiler tool has an impact on activities and work products (checking source code, producing binary code), just as the developer role has (checking design models, producing source code). If almost every development team uses a compiler, only a small percentage still use case tools, or test tools, or configuration management tools. Each kind of tool needs to be represented in the process, accompanied by its influence on the organization.

9.5 Development rules and guidelines

Development rules and guidelines are based on the technologies used, on the developer's current points of view (introduced later - analysis, development, building a prototype, etc.) and on the kinds of expected work products.

They describe the skill of a development organization, by detailing all rules and guidelines that have to be followed for producing every work product. They ensure that a development process is repeatable, using a growing level of quality. They constitute a development knowledge repository for the developers, and therefore guarantee that the development results do not rely too heavily on someone's particular performance that may not be reproducible.

Possible examples of development rules or guidelines are :

- Document templates, that describe how a particular kind of document should be made, what chapters should be included, what information should be in these chapters,
- Development guidelines in a specific programming language, such as naming rules, sources organization rules, the source structuring rule, and language features that are encouraged to be used or prohibited,
- Specific modeling consistency rules, that conduct the modeling task and help declaring a model correct or not. These rules can enforce maintainability or readability or robustness or efficiency, depending on the current development point of view.
- Design patterns, that represent design knowledge in particular, and that can be recommended in certain contexts or prohibited,
- Translation rules, such as model to code translation, or model to database schema translation,
- Metrics, that are expressed by formula in order to measure different aspects such as complexity, maintainability, encapsulation, performance, for several kinds of work products.
- policies such as configuration management policies, bug management policies, recruitment policies, quality control policies
- techniques customizations, such as UML subsets, UML customizations, design patterns

10. Properties that this language should have

10.1 Introduction

The system/process definition language needs to organize the process elements, in order to ensure the following properties :

- structuring mechanism : work products, activities, rules and any other process components must be structured into consistent sets of elements, deliverables that can be distributed, shared, customized, etc.
- extensibility : many users are not willing to redefine from scratch a new methodology for their projects. They cannot take other methodologies as they are either. They need to adapt already defined methodologies, in order to target their specific constraints, culture, etc. This is the reason why the deliverables must be customizable and fulfill the extensibility property.
- relationships to techniques : imagine a technique, UML for example, many processes can be applied on it. As we want to automate processes, we need both a close connection between

processes and techniques, and an interchangeable connection. Techniques may be differently customized depending on the process definition combinations that are applied.

- Parameterization mechanisms : process definition can define parameters, that may have different values depending on the project context. Thus, the project definition defines a project variability range, on which it can be applied as well. Metrics, automated code generation, rules, may for example depend on general parameters such as the project size, or on physical parameters such as directory paths.

10.2 Organizing process engineering components

There are two fundamental questions that have to be answered in order to develop a software methodology framework.

- How can these numerous process engineering components be organized into consistent, reusable and customizable groups ?
- How should the methodology/technology mapping be managed ?

As stated before, process engineering is much more related to " how to use " a technology than to the technology itself. There is a kind of " orthogonal " relationship between these different aspects. For example, the development phases definition does not depend on the programming language used. However, their precise content is influenced by the technique. We can say the same process can be applied on different techniques, though there may exist some adaptations, and that the same technique can be used by different processes, though they may also need some customizations.

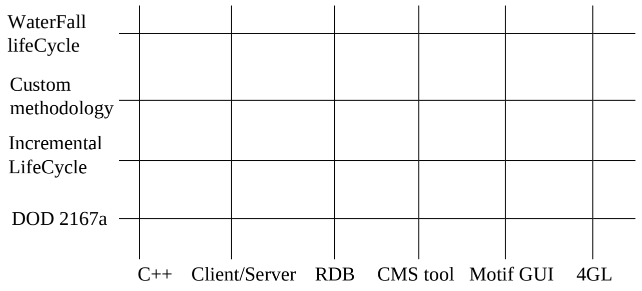


Figure A-2 - *Methodologies and technology are by nature " orthogonal "*

Figure 2 presents that orthogonal aspect. Given a particular methodology (say Waterfall) and a set of technologies to use (say C++ and UML), the requirement consists in deriving a set of predefined " process engineering components groups " in order to obtain a customized methodology which is " Waterfall for UML with C++ ".

Obviously, that customized methodology will be further customized in order to define the " Waterfall for UML with C++ dedicated to my organization " methodology. This new customized methodology needs to be packaged so that it can be itself distributed, shared, improved, etc.
There is a specific structuring mechanism that has to be used for this purpose. We call it " *Point-of-View* ".

10.3 Introducing the " Point-of-View " mechanism

As seen above, process engineering and technologies have a orthogonal relation. This is for example the case of UML technology. In this case, depending on the methodology used, there may exist :

- different stereotypes, different tagged values, which provide means to adapt UML to different domain notations, different methodologies, or different underlying technologies,
- different document generation templates, that may produce different documentation from the same model
- different code generators, that may use different programming conventions, or that may use different target combinations (C++/SQL, Java/CORBA, etc.)
- different design patterns, that promote different kinds of design approaches
- different modeling rules, that heavily depend on methodologies, technologies,
- different configuration management policies, that manage links between generated code, models, documentation, etc.

Thus, considering a given UML model, there may exist different usage of it, depending on the point of view being used : it may be differently interpreted during analysis or design, for real time purposes or for data storage purposes, for a C++ development , or for a Java development.

Many elements will change : the software products that have to be produced, the rules that have to be abided by, the necessary tagged values, etc. The understanding of the model will not be the same, and the functions that a tool may provide are not the same either.

We call " point-of-view " the perspective from which the model is considered. Thus, process engineering defines a set of points-of-view that may be applied to a model, each of them providing the necessary set of " process engineering elements " required. Points-of-view can materialize development phases, or more accurately development activity types such as " analysis ", or data schema definition, C++ programming, data schema optimization, etc.

The " point-of-view " concept is orthogonal to the " software element " one : the same viewpoint can be applied on different software elements, whereas different points-of-view can be applied on the same software element. Point-of-view is a major concept for structuring the process engineering concepts.

RM-ODP has the viewpoint concept, which is similar, but it is different from the point-of-view concept described here.   As for the RM-ODP viewpoint concept, the point-of-view notion defines a particular angle for viewing a problem. However, the point-of-view notion is dedicated to a stabilized meta-model (different angles for considering a meta-model); whereas the point-of-view notion is not dedicated to any specific model. In addition, the point-of-view notion does not provide predefined points of view, and is extensible to any number of kinds of point of views, whereas RM-ODP viewpoints are limited to five predefined kinds of viewpoints. Points-of-view must be customizable by users, in order to let them take elements from a " process engineering repository " and adapt them to their needs. That is why inheritance links between points-of-view may exist.

Point-of-View is a mechanism for structuring rules, that provides a means to look at the same meta-class from different angles of view, depending on the expected meta-model usage. For example, documentation generation rules, C++ patterns rules, relational database rules, will each have a specific interest in the meta-model, materialized into the so called “ point-of-view ” concept.

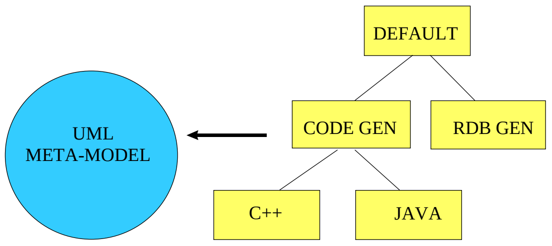
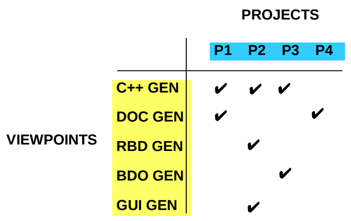


Figure A-3 : *Different points-of-view consider the same meta-model from different target perspectives*

The point-of-view structure may be a hierarchical structure, showing refinement links between different points-of-view.

Points-of-view are very important for the developers, but are necessary for the final users too : at the project level, users choose which already defined point-of-view they want. The project becomes customized by the user’s choice for their specific requirement, having specific rules and generation dedicated to their need (RDB, ODB, Client/Server, etc.).



		PROJECTS			
		P1	P2	P3	P4
VIEWPOINTS	C++ GEN	✔	✔	✔	
	DOC GEN	✔			✔
	RBD GEN		✔		
	BDO GEN			✔	
	GUI GEN		✔		

Figure A-4 - *Every project can choose its specific points-of-view.*

Methodologies can be customized by selecting and refining a set of points-of-view that already exist. Tools may change their consistency control mode, functionality, the views they present to the user. They adapt their behavior to a specific technique such as UML, to the selected processes the points-of-view represent.

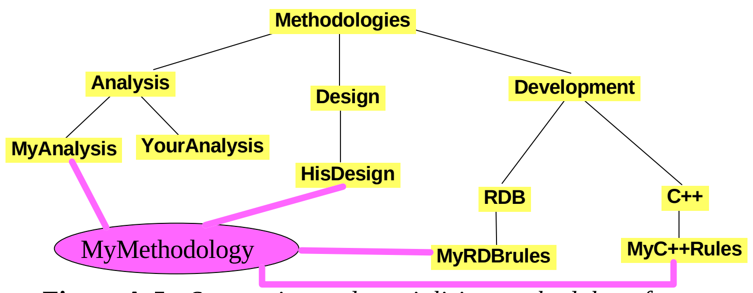


Figure A-5 - *Structuring and specializing methodology fragments*

Appendix B - Example an SPDL Meta-Model

Note: Tthis SPDL meta-model is just a proof-of-concept. Such a meta-model could exist and it does not necessarily reflect any meta-model that could be adopted by the ADTF as a result of a future RFP. It is for illustrative purposes only, to clarify what the meta-model of a SPDL could look like.

11. Logical Model

11.1 Actors and Use Cases

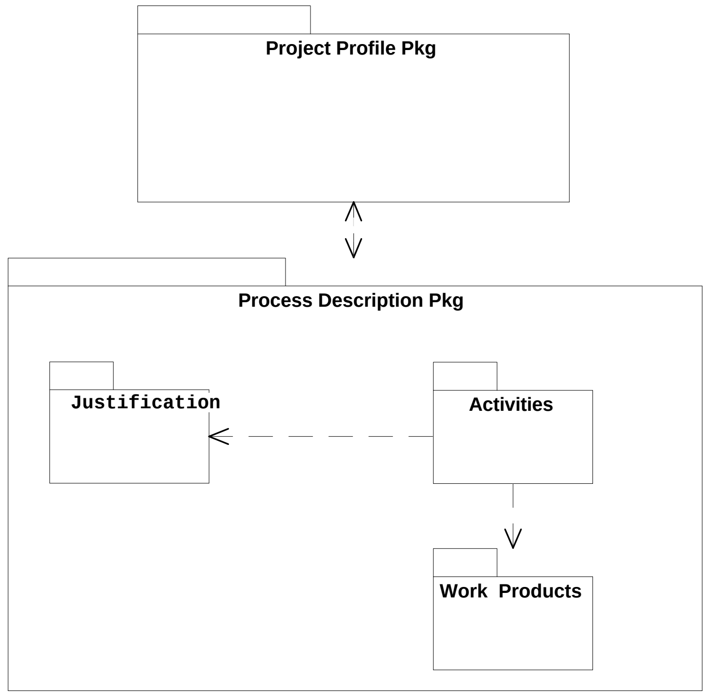
Actors	Uses
Process Engineer Methodologist Author	To define, express, communicate, and justify process and process components at all levels and their associated goals and risks, including suitability for target organizations and projects. Process and Process Components include • process • OO method/methodology (such as OMT, Fusion, Lockheed Martin ACC Method Framework) • procedure • development patterns • Software Development Life Cycles (e.g., waterfall, spiral, iterative/incremental, fountain, scrum) • lessons learned
Process Engineer	To compare/contrast different processes and process components, at all levels based on suitability to organization and project circumstances and alignment to risks and goals.
Program Manager	To modify, tailor (both a priori and dynamically) processes and process components by adding, deleting, or modifying activities, their applied efforts and resources, and their entrance and exit criteria in response to changing priorities, and circumstances.
Tool Vendor	To communicate, exchange, and depict models of process and process components at all levels.
Instructors Methodologist	To use as a basis for instruction for process and process components that is: 1) capable of being abstracted for the level of audience 2) easily tailorable to students' priorities and circumstances
Program Manager	To determine how to assign resources, order activities, evaluate progress, assess risks

Developer	To determine proper steps and actions, warn against potential problems.

12. Package Level

12.1 Model Overview

The diagram gives the high-level over-view of the model.



12.1.1

FigureB-1 Package Diagram

12.2 Project Profile Package

This package exists to allow:

- Process/Methodology authors to describe the types of organizations and projects for which their process or method is suitable.
- Evaluating organizations and project teams to select the Process/Methodology that is suitable for their situation.

Since the potential criteria for characterizing organization and projects are numerous and evolving, the scheme should not only allow for standard categories of criteria, but should be dynamically adaptable, extendible, and sub-divisible.

12.3 Process Description Package

This package contains the description of the activities and tasks of the process or method (Package Activities), the reasons why and under what circumstances these Activities are done (Package Purposes) and the products (Package Work Products) produced by these activities.

Though we use the term "Process" in the package name, it is intended that system development activities of any scale can be captured. This would include large-scale System Development Activities such as (but not limited to):

- OO Methods (e.g., OMT, Booch, Fusion, Rational Objectory, Process, Lockheed Martin ACC Development Framework)
- System/Software Development Life Cycles (e.g., waterfall, spiral, fountain, scrum)
- Non-OO methods (e.g., Method 1)

- It would also include smaller scale descriptions of activities taken to address particular risks. These include:
- Managerial Patterns
- Lessons Learned
- Individual Project Plans

Essentially, this is a Software Development Process Language (SPDL) that allows us to model and describe process solutions in the system development domain.

12.3.1 Justification

Every activity must be justified. An activity is only performed for a purpose, either to achieve goals or to mitigate risks.

Processes with justified steps are easier to teach, learn, understand, and use. Without justification, it could be said that no one really "knows" the process.

If the activities are justified, then when circumstance change, the activities may be modified appropriately. Under a different set of circumstances, an Activity may need to be shortened, changed, de-emphasized, or even eliminated.

The Justification Package is also closely connected to the Project Profile Package. A given organization, for a given project, will have a set of perceived goals and risks. They need the ability to compare their goals and risks to the Process Description's Justifications to choose the best process. Then, by contrasting them, a development team should be able to modify/tailor the Activities to best meet their goals and risks. And as development proceeds, and new risks appear, the team needs to be able to dynamically tailor the Activities.

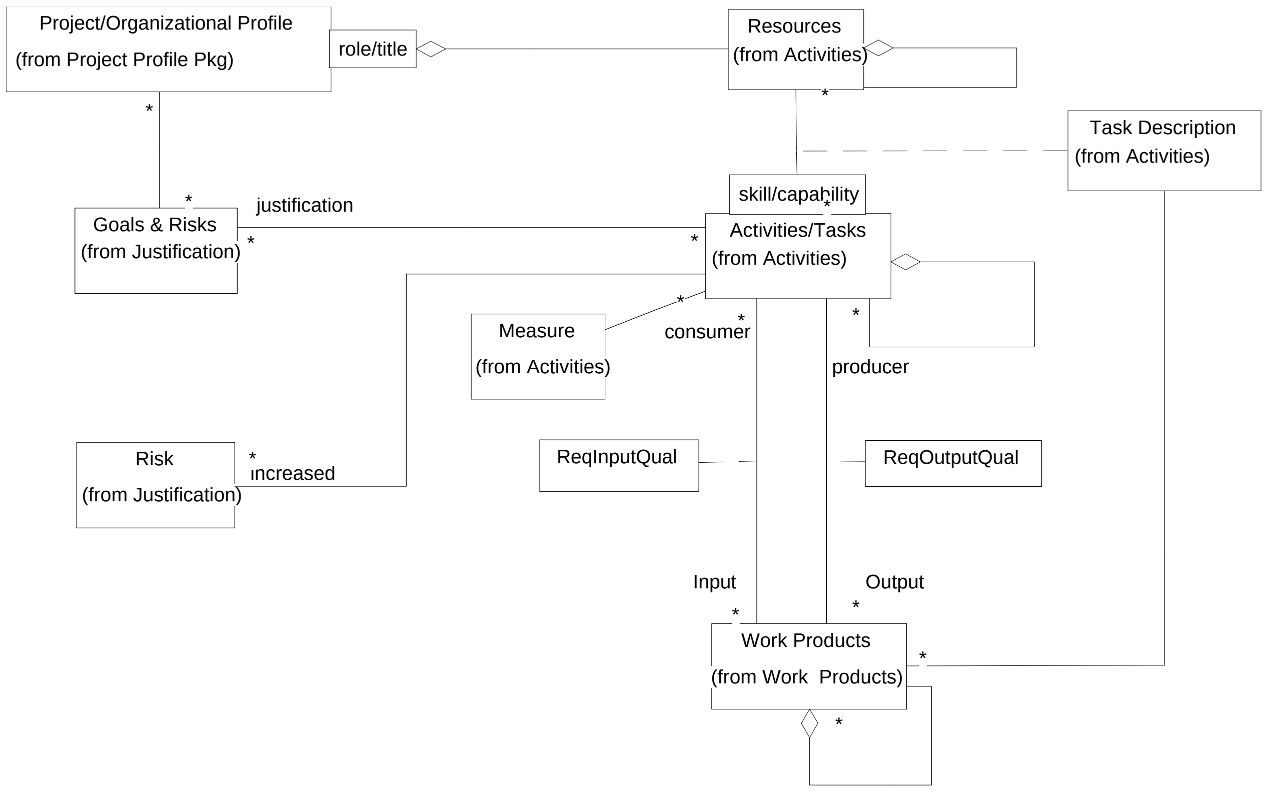
12.3.2 Activities

This package contains the descriptions of the activities, tasks, steps, their resources, and metrics.

12.3.3 Work Products

This package contains the items produced for each activity performed.

13. High-Level Static Diagram



FigureB-2 High-Level Static Diagram

13.1 Project Profile Package

13.1.1 Project/Organizational Profile Class

This represents the set of information needed to profile a particular organization and a particular project.

At least the following organizational characteristics should be accounted for:

- Industry
- Size
- Formality
- SEI Level

At least the following project characteristics should be accounted for:

- Size
- Budget Profile (e.g., Cost, Effort, Time)
- Formality
- Implementation Profile (e.g., Architecture, Language, DB)

13.1.1.1 Relationships

13.1.1.1.1 To Resources

- A Project/Organization owns certain resources.
- Each project may call the equivalent resource by a different title depending on the role the resource place on the project and depending on the role nomenclature chosen by the project.
- Note: The resources required by a particular Activity may not be available as part of a project.

13.1.1.1.2 To Goals or Risks

- A project has a set of goals to achieve and a set of risks to avoid.
- An organization may have a set of Goals and Risks that are common to many of its projects
- These Goals and Risks may be for many projects and many organizations.

13.2 Justification Package

13.2.1 Goals & Risks Class

A Project is started because an organization has a set or goals that they need to reach or alternatively, a set of risks that they need to address. One way of looking at this is to consider that the Goals & Risks are the requirements that a particular set of Activities/Tasks are undertaken to solve.

There will need to be at least two versions of this class in the final model. One to represent the Goals & Risks needed to justify a particular Activity/Task, the other to represent the Goals & Risks actually occurred for a particular Project.

13.2.1.1 Relationships

13.2.1.1.1 To Project/Organization Profile

- These Goals and Risks may be for many projects and many organizations.
- Each Activity (of whatever level) is performed to achieve some goals and to mitigate some risks.
- The set of Goals and Risks act as justification for an activity
- Several activities may be justified by the same Goal or Risk.

13.2.1.1.2 To Activities/Tasks

- A project has a set of goals to achieve and a set of risks to avoid.
- An organization may a set of Goals and Risks that are common to many of its projects
- These Goals and Risks may be for many projects and many organizations.

13.2.1.2 Subclasses: Risk

Risks may be of many types.

- **Business:** e.g., Competitive need to have properly positioned project by a certain date, Organization needs particular capability to improve customer retention
- **Managerial:** e.g., Cost, schedule, people,
- **Technical** (Solution): e.g., will this solution work, be fast enough, interoperate.
- **Developmental:** Can we (with our organization, skills, discipline), develop this project. Can we reach consensus, avoid analysis paralysis, requirements creep. Can we "design-to-cost"?…

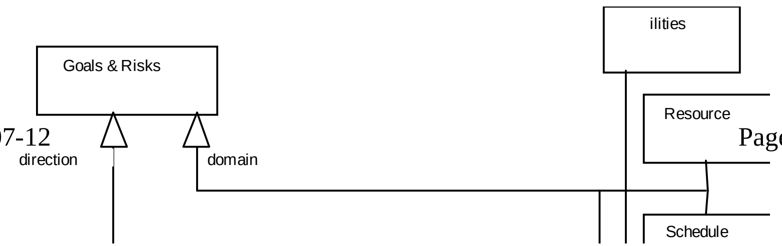


Figure B-3 - Types of Goals/Risks

13.2.1.3 Relationships

13.2.1.3.1 To Activities/Tasks

- Every activity has a set of risks that they may increase. These can be considered the "pitfalls" to watch out for when doing the activity.
- Some risks can be increased while doing several different activities.

13.3 Activities Package

13.3.1 Resources Class

The Resources Class represents the people and other enabling resources (e.g, development platform, tools, databases, access to equipment) that are used to perform a particular activity or task.

There will need to be at least two versions of this class in the final model. One to represent the skills/capabilities needed to perform a particular Activity/Task, the other to represent the skills and capabilities available to the Project.

13.3.1.1 Relationships

13.3.1.1.1 To Project/Organization Profile

- A Project/Organization owns certain resources. Each project may call the equivalent resource by a different title depending on the role the resource place on the project and depending on the role nomenclature chosen by the project.
- Note that the resources required by a particular Activity may not be available as part of a project.

13.3.1.1.2 To Activities/Tasks

- For a particular Activity/Task, there are a set of resources need to perform that task. These resources may be partitioned by differing skills and capability descriptions.
- Each resource may be used on many different Activities/Tasks.
- For each resource participating in an activity, there are Task Descriptions (or Instructions) on how to apply that resource for this task.

13.3.1.1.3 To Resources

- A Resource can be at several different levels, for example, a Database Team is a resource that might contain a some tuning specialists for different databases, a manager, a tester....

13.3.1.2 Subclasses

See figure 4 for some of the different types of Resources that need to accounted for.

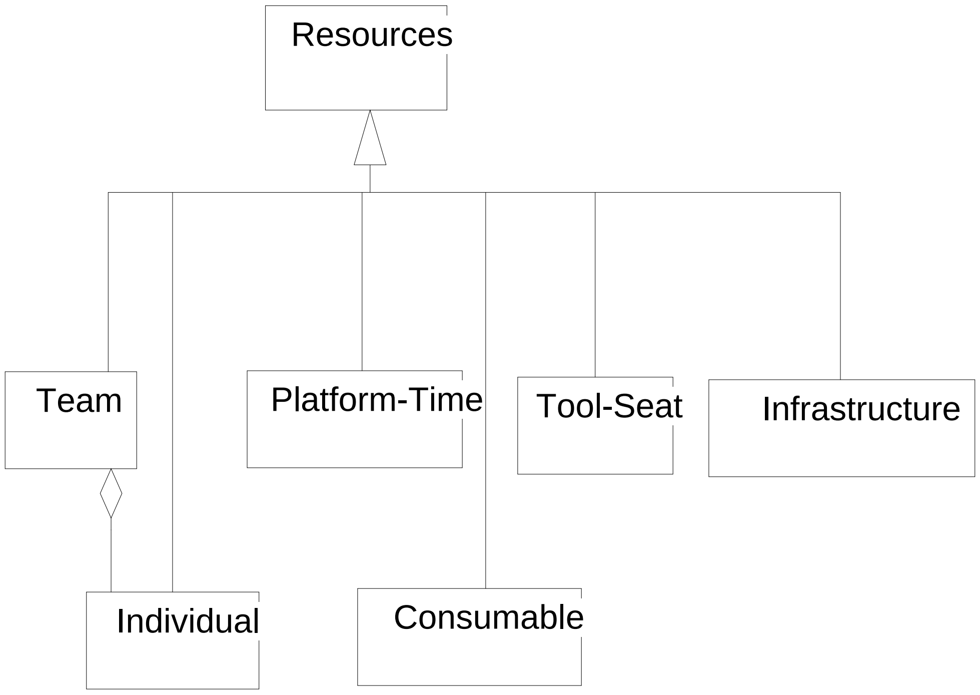


Figure 3 Types of Resources

13.3.2 Task Description Class

A Task Description is the instructions given to a particular (human) resource to perform a given Activity/Task on the Work Products of the Task. For non human resources (such a hardware platform), it is the description of how that resource will be used on the given Activity/Task to product the Work Products of the Task.

13.3.2.1 Relationships

13.3.2.1.1 To Work Products

- For each Resource assigned to a Task, a Task Description is needed to capture how that Resource will be applied to a particular Work Product

- Usually, a Task Description will cover only one Work Product, or may have sections for multiple work products.
- In addition, while we normally think of a Task Description as being needed to cover how a Work Product will be produced, it is possible that a Task Description will be needed to incorporate an input Work Product

13.3.3 Activities/Tasks Class

The Activities / Tasks class covers the range of applicable steps of development, from large-scale activities such as Analysis, through smaller activities, such as code walk-through. It also includes management reviews, steps from the method....

Activities include things such as; Developments, Measure, Manage, Buy, Assess, or Decide.

Activities can have sub-activities at different scales. Whole nomenclatures for the different levels are used, but they usually hide the essential similarity of the levels.

Each Activity/Task has some justification, either goals to achieve and/ or risks to abate, and will have it's own sets of risks that may occur if it is performed.

An Activity/Task often has a set of inputs that must at least meet a required level of quality before the activity may start. It also has a set of outputs that must at least meet a required level of quality before the activity can be considered completed. Note that an output may be (and usually is) usable by a following activity even before it is "finished" by the originating activity. In addition, a Work Product, in different states, may be input and output to the same activity.

13.3.3.1 Relationships

13.3.3.1.1 To Resources

- For a particular Activity/Task, there are a set of resources need to perform that task. These resources may be partitioned by differing skills and capability descriptions.
- Each resource may be used on many different Activities/Tasks.
- For each resource participating in an activity, there are Task Descriptions (or Instructions) on how to apply that resource for this task.

13.3.3.1.2 To Goals & Risks

- These Goals and Risks may be for many projects and many organizations.
- Each Activity (of whatever level) is performed to achieve some goals and to mitigate some risks.
- The set of Goals and Risks act as justification for an activity
- Several activities may be justified by the same Goal or Risk

13.3.3.1.3 To Risks

- Every activity has a set of risks that they may increase. These can be considered the "pitfalls" to watch out for when doing the activity.
- Some risks can be increased while doing several different activities.

13.3.3.1.4 To Measure

- Each Activity or Task will have a set of measures (often called Metrics), to determine if the Activity/Task:
  a) is on Track (schedule/cost)
  b) is of sufficient quality

13.3.3.1.5 To Work Products

- An Activity/Task often has a set of inputs that must at least meet a required level of quality before the activity may start. It also has a set of outputs that must at least meet a required level of quality before the activity can be considered completed. Note that an output may be (and usually is) usable by a following activity even before it is "finished" by the originating activity. In addition, a Work Product, in different states, may be input and output to the same activity.

13.3.3.1.6 To Activities/Tasks

- Activities can have sub-activities at different scales. Whole nomenclatures for the different levels are used, but they usually hide the essential similarity of the levels.

13.3.3.2 Subclasses

13.3.3.2.1 Measure

Each Activity or Task will have a set of measures (often called Metrics), to determine if the Activity/Task:
a) is on Track (schedule/cost)
b) is of sufficient quality

A Measure in this context is actually an Activity/Task by itself, as it is equivalent to some activity being performed. A Measure must be independently justified (is it worth to collect this), may cause its own risk (developers design differently depending on the metric) and has its own set of required inputs and outputs. A Measure may have lower-level component Measures and may be a component of a larger Measure. To perform a particular Measure, we also need resources.
A Measure requires at least one output Work Product with a Required Output Quality that is a range of acceptable values for that measure.

For each Measure, they will need to be equivalent class(s) in the Organization/Project Profile Package that describe historical results for that organization/project.

13.4 Work Products Package

13.4.1 Work Product Class

The Work Products class holds all the produced physical and conceptual products. It includes documents, decisions, diagrams, figures of merit, measures, etc. Work Products can be complex and contain within them other work products. For example, a Requirements Document would contain a Use Case Model, which would in turn contain, a Use Case Overview, which in turn would contain a list of Actors. It is only necessary to identify those Work Products that require separate Activities/Tasks or are produced by different Resources. *Remember that the Work Products may be both input and output to same Activity/Task and may have multiple states of completion.*

13.4.1.1 Relationships

13.4.1.1.1 To Activities/Tasks

- An Activity/Task often has a set of inputs that must at least meet a required level of quality before the activity may start. It also has a set of outputs that must at least meet a required level of quality before the activity can be considered completed. Note that an output may be (and usually is) usable by a following activity even before it is "finished" by the originating activity. In addition, a Work Product, in different states, may be input and output to the same activity.

13.4.1.1.2 To Task Description

- For each Resource assigned to a Task, a Task Description is needed to capture how that Resource will be applied to a particular Work Product
- Usually, a Task Description will cover only one Work Product, or may have sections for multiple work products.
- In addition, while we normally think of a Task Description as being needed to cover how a Work Product will be produced, it is possible that a Task Description will be needed to incorporate an input Work Product

13.4.1.2 Subclasses

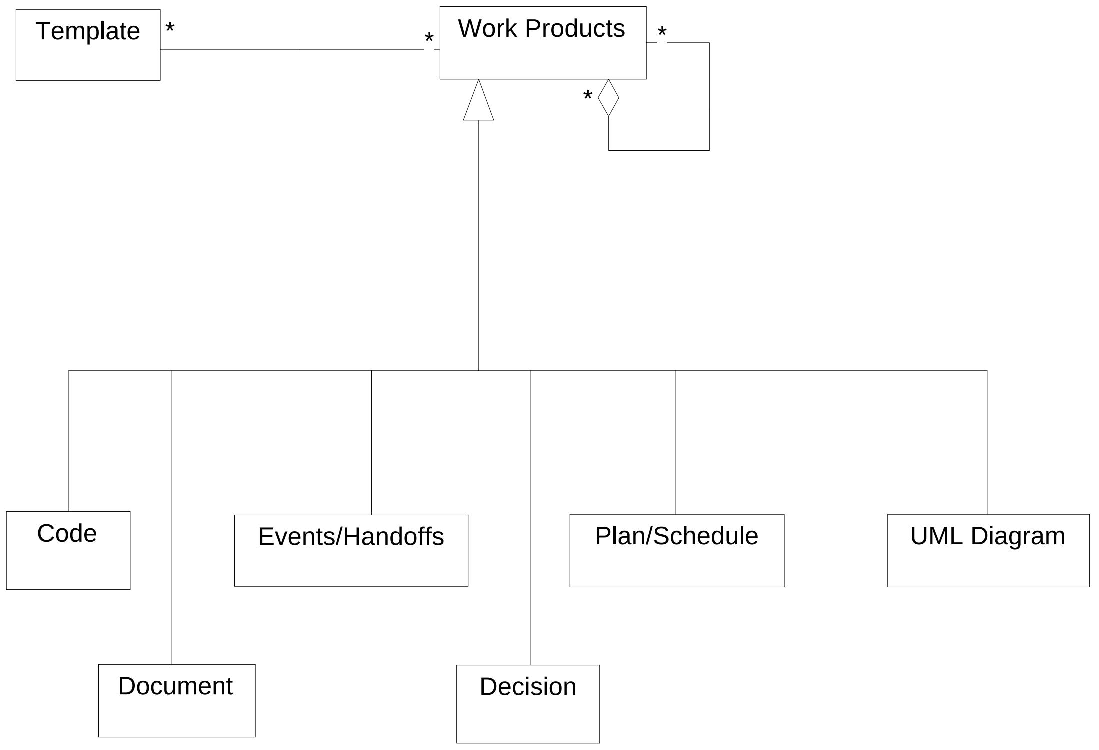
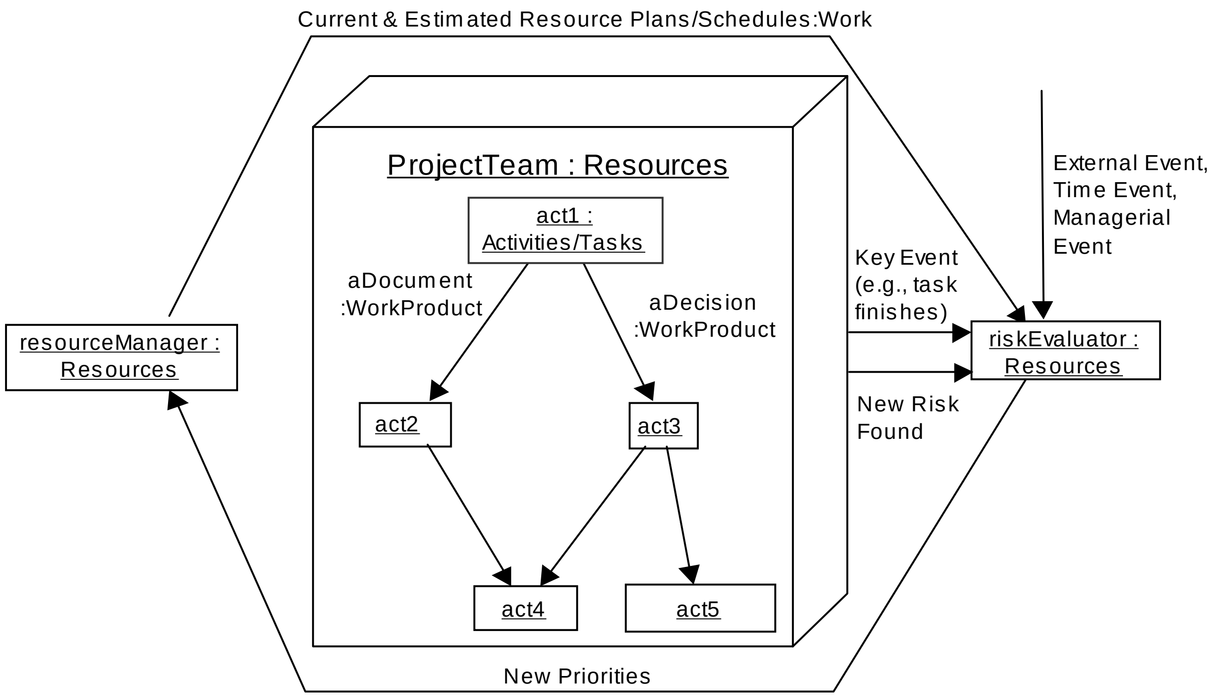


Figure B-5 Types of Work Products

14. Dynamic Model


Current & Estimated Resource Plans/Schedules:Work
ProjectTeam : Resources
act1 : Activities/Tasks
aDocument :WorkProduct
aDecision :WorkProduct
act2
act3
act4
act5
resourceManager : Resources
riskEvaluator : Resources
External Event, Time Event, Managerial Event
Key Event (e.g., task finishes)
New Risk Found
New Priorities

The dynamic model for arranging the activities of a system development effort is not a workflow model. Workflow models are primarily have activities that need to complete before the following activities start. In modern system development, completion is not the most common criteria for starting the next step.

Completion of an activity/task is not typically used as the criteria for several reasons:

- Some tasks are nearly impossible to prove complete. If completing test implies that there are no undetected bugs, or if completing analysis implies that there are no missing requirements, then it impossible to prove and nearly impossible to detect.
- The effort to "complete" a task typically follows the 90/10 rule. The last 10% needed to complete a task takes 90% of the effort. Therefore, even if completion is detectable, it may be very expensive. The value to the project for completing the task may be less than the course of doing the task.
- Shortened time lines for project completion (e.g., time-to-market pressures), cause overlap of tasks. Follow-on tasks using the output a of a task, can often start as soon as part of the output is complete. This is the basis for incremental development.

Instead of a workflow model, we wish to model development as set of cooperative jobs executing on a computer but that competing for resources.

14.1 Activity/Task as a Computer Job

Each Activity/Task (from the Activities Package) is modeled as computer job using some of the available system resources (Resources from the Activities Package). A component Activity/Task is a sub task of the computer job. Input and Output Work Products (from the Work Products Package) are ongoing flows between jobs. The processor is really an aggregate of the available resources.

14.2 Task Priority

The priority for each job is set based on the relative risk that the Activity/Task addresses. These priorities are initially set before the project starts based on the Organizational/Project Profile and their associated Goals/Risk (from the Justification Package). For example, the priority of the testing Activity is different for shrink-wrapped software than for medical equipment software.

14.2.1 Risk Evaluator

There is a special resource whose job it is to evaluate and assign the relative priorities of the tasks. The Risk Evaluator is typically a technical or engineering responsibility

14.3 Resource Assignment

Corresponding to the Priority is the allocation of resources. Resources are allocated in proportion to the priority, subject to availability and other constraints.

14.3.1 Resource Assigner

There is a special resource whose job it is to assign the resources to the appropriate jobs based on their relative priorities. The Resource Assigner job is typically a managerial responsibility. This may be the same people who are the Risk Evaluations, but their jobs are logically distinct.

14.4 Dynamic Priority

System development needs the ability to dynamically change the priorities of these tasks. In the simplest form, if a task is not yet able to run, the priority for a task is set to zero (and the corresponding burn-rate for resources is set to zero). In addition, tasks that have finished also have their priorities lowered, and their resources taken away. Under many circumstances the task is never fully stopped, if a proper appears later in the development, an earlier task may be reopened (priority-boosted).

More farsighted development projects need to dynamically change their priorities for other reasons. While it is true that Tasks that address the most serious problems run at the higher priorities, but may be starved for inputs. If there is a problem in an earlier task, it may be necessary to assign more resources to that phases and to take such resources from downstream tasks.

Whole tasks may need to be scheduled if during the development a new area of risk becomes apparent. For example, performance may not be thought of as a risk area until it appears that there is a bottleneck in processing.

- The criteria for re-evaluation are part of the methodology/process, but typically include:
- Starvation of an Activity/Task,
- Completion of an Activity/Task,
- Missing a schedule date
- Finding a new problem (risk)
- Changing resource availability
- Major design decision

15. Dynamics Sample

The following sample of the dynamics is given to illustrate that the model accurately describes practical project management and is suitable for our purposes.

In our VSSM (Very Simple Sample Methodology) we have three Activities, Use Case, Analysis, and Development. Each Activity has an output document and several diagrams. It has been determined that the three Activities address the following Risks (Use Case – matching user needs; Analysis -- good understanding of requirements; Development -- producing a timely system). As a whole, the VSSM approach is only suitable for programs using Java for non-critical development.

In VSSM, the Analysis activity has two sub activities, Domain Analysis and Application Analysis. Domain Analysis and Use Case Analysis are started at the same time. Domain Analysis exists to address the risk that the developers are unfamiliar with the domain, and Application Analysis exists to address the risk that the developers are unfamiliar with the application.

The VSSO (Very Simple Sample Organization) checks through the set of possible methodologies available and chooses VSSM as most closely matching their size, temperament, and experience. The VSSO Process Engineering Team, tailor the VSSM to increase the priority given to the Use Case Activity -- they feel that VSSO projects require more attention to user input than the pre-defined VSSM has. In addition, since VSSO has littler turn-over of resources, their developers know the domain well – allowing them to de-emphasize domain analysis. They call this new method, VSSM'.

As now defined, in VSSM' both Analysis and Development use the outputs of the Use Case Activity. The domain analysis activity is de-emphasized and combined with application analysis. The outputs of Analysis are used by Development.

There are not sufficient resources for each of these steps in the VSSO Organizational Profile to be done simultaneously, but the available skills are sufficient. In addition, the time-line is very short.

Because of the VSSM' emphasis on user input, and on the unavailability of the required inputs for the later phases, the Project starts with the Use Case Activity. This Activity logically also has the higher priority, as without understanding the user needs, any produced project would not be useful.

As the Use Case Activity progresses, the risk of not-understanding the user's needs declines. And as time wears on, the risk that the project doesn't yet understand the requirements increases. In addition, the rate of return for the Use Case Activity declines. That is, the effort to find the n+1 use case becomes more expensive. At some point, it becomes more valuable to the project to move resources from the Use Case Activity to the Analysis activity and increase the Analysis activity's relative priority. The System Engineer (Risk Evaluator) tells the Project Manager (Resource Manager) of the change to priority. The Project Manager determines the appropriate people to free up from the Use Case Activity and moves them to the Analysis Activity.

The Analysis Activity then starts up, using some of the early outputs of the Use Case Activity. Analysis progress, problems with the analysis may appear that are traced to missing use cases. As each one of these occur, the System Engineer (Risk Evaluator) determines whether these missing use cases need to be evaluated by the Use Case Activity. If they do, the Project manager (resource Manager) determines what changes to resources and schedule are needed.

As progress slows down in the Use Case activity, and less use case are being found in later activities, it may become useful to stop them completely and release their resources. If analysis is sufficiently along, and the appropriate skills are available, these resources may be transferred to the Development activity.

If later development still finds more unknown use cases, the Risk Evaluator determines the least risky approach; a delayed delivery for the new use case, re-opening the Use Case activity, or treating the use case where it is found.