Grant agreement no.: *100012*

D6.3-C Analysis of Standard Process Models

Author(s): **Jesús Benedicto** **ATOS**
Igor Rosenberg **ATOS**
Ignacio Soler **ATOS**
Naia Arana **ESI**
Huascar Espinoza **ESI**

Issue Date	October 2010 (m21)
Deliverable Number	D6.3-C
WP Number	WP6: Compositional Verification, Validation and Certification
Status	Delivered

Dissemination level	
X	**PU** = Public
	PP = Restricted to other programme participants (including the JU)
	RE = Restricted to a group specified by the consortium (including the JU)
	CO = Confidential, only for members of the consortium (including the JU)

Document history

V	Date	Author	Description
0.1	*2010-04-27*	*ESI*	*ToC*
0.2	*2010-08-15*	*ATOS*	*Updated ToC structure. Added contents to chapter 1. and chapter 2.*
0.3	*September 2010*	*ESI*	*Updated Introduction and Conclusions. Chapters review*
0.4	*October 2010*	*ATOS*	*Chapters improvement, final edition and Internal Review.*

Summary

The D6.3-C Analysis of Standard Process Models is a public document delivered in the context of WP6, Task 6.3: Specification of a Certification Metamodel for Energy Management Deployment. Task 6.3 aims at the definition of a prototype that will help assessing future eDIANA project/product/component implementations against relevant standards in the domain of embedded systems for energy efficiency applcaitions (see deliverable D6.3-A for a full descriptions of relevant standards), good practices, and eDIANA recommendations.

This document deals with evaluating existing process modelling notations and metamodels in order to establish a cohesive connection of development process instances with the Certification Metamodel proposed in T6.3.

Contents

Abbreviations

ARIS-EPC	ARIS Event-Driven Process Chain
BPDM	Business Process Definition MetaModel
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
BPMN	Business Process Modelling Notation
BPMI	Business Process Management Initiative
BPML	Business Process Modelling Language (BPML)
ebXML	Electronic Bussiness XML
eDIANA	Embedded Systems for Energy Efficient Buildings
IDEF	ICAM Definition Language
IT	Information Technologies
MOF	Meta-Object Facility
OMG	Object Management Group
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPEM	Software Process Engineering Metamodel
UDDI	Universal Description, Discovery and Integration
UML	Unified Modelling Language
WBS	Work Breakdown Structures
WfMC	Workflow Management Coalition
WSDL	Web Service Description Language

WSFL	Web Services Flow Language
XPDL	XML Workflow Definition Language
XMI	XML Metadata Interchange

Table of Figures

Table of Tables

1. Introduction

This document is the third document of Task 6.3, "Specification of a Certification Metamodel for Energy management Deployments", in WP6, "Compositional Verification, Validation and Certification". Task 6.1 and 6.2 focus on Verification and Validation issues, while Task 6.3 deals with Certification aspects.

As it has been mentioned in previous deliverables, certification is one of the most time consuming tasks to develop embedded system products toward certain regulations (external as well as internal ones) and is usually done for the products when the life cycle has finished.

Certification involves monitoring the development process, which is usually done manually by high level roles having a full overview by compiling information from many people involved in the process and a lot of dependencies. There are a number of gaps in this area that are targeted by T6.3 and this deliverable in particular:

- Lack of infrastructure for software development, and support tooling dedicated to certification.
- Time consuming mechanism/process to perform legal certification and 'internal certification' (i.e. qualification).

Previous outcomes of this task have described and analyzed some relevant standards, best practices and/or regulations that are applicable to the eDIANA domain. Some of those standards, such as for example IEC 61508 or CMMI, cover the lifecycle with all their phases, or define a methodology that must be implemented. On the other hand, other standards apply on the resultant components or products and their final behavior, measurements, etc. EPBD (Energy Performance of Buildings Directive) is an example of this kind of standard. EPBD is oriented to certifying the building performance as a final product, and not the process to build this final product.

Due to this diversity of standards, the prototype targeted in eDIANA (T6.3) focuses on creating a general certification "language" by means of a structured semi-formal metamodel, which will act like a template for certification requirements specification. This metamodel will be used to build domain-specific libraries of certification models, which will act as a knowledge database, providing information about eDIANA-related standards (IEC 61508, EPBD, etc.) and other eDIANA-specific implementation requirements (some kind of eDIANA-compliant label). Such "knowledge database" is used to build a set of guidelines akin to "spell-checking", in which a number of compliance checks are performed to assess the degree of compliance of embedded system products against eDIANA-related standards.

However, in most businesses and companies, the definition of the internal processes, private guidelines, or methodology is an essential issue to the satisfactory achievement of their objectives. This leads our work to study possible future interactions of the Certification prototype with engineering process modeling. This possible connection may be through extensions of the prototype to connect it to process modeling tools, as well as extensions of the prototype to add process modeling functionalities.

This deliverable surveys and assesses process languages to discover the possibilities to extend the Certification prototype. This includes the study of business process modeling notations and Business Process Modeling languages such as:

- Software & Systems Process Engineering Metamodel (SPEM)
- Object-oriented Process, Environment, and Notation (OPEN)
- Business Process Definition Metamodel (BPDM)
- Unified Modeling Language (UML) 2.0 Activity Diagram
- ARIS - Event-Driven Process Chain (EPC)
- Business Process Management Notation (BPMN)
- XML Process Definition Language (XPDL)
- Business Process Execution Language (BPEL).

The depth of the analysis is closely linked to the information available for each of the metamodels or languages analyzed, and with respect to the market share.

1.1 Terms and Definitions

Business Model: A Business Model is a conceptual tool containing a set of objects, concepts and their relationships with the objective to express the business logic of a specific firm. It is a description of the value a company offers to one or several segments or customers and of the architecture of the firm and its network of partners for creating, marketing and delivering this value and relationship capital, to generate profitable and sustainable revenue streams [1]

Business Process: A business process is a sequence of activities triggered by a certain input that results in a valuable output. Business Process Management is about analyzing those activities in a structured way and eventually supporting their execution with a workflow application

Business Reference Model: A Business Reference Model defines a canonical set of process areas which group business processes according to the primary business function. A process area includes a sequence of processes that are combined to form the value chain of the studied business area. Examples for business reference models are SCOR, VCOR, eTOM or ITIL®.

Meta-Model: A Meta-Model is a model of a modelling language. Applying language therory for levelling languages, the result is a hierarchy of languages, meta-languages, etc. The creation of a Meta-Model is also done by using a modelling language. The model defining the meta-modelling language is the meta-meta model or meta2-model [2].

Model: A Model is an abstract graph-based description of a complex real-world system.

Modelling Method / Language: The language for representing a model is described by its meta-model. The concepts of a Modelling method are described by elements such as classes, relationships, attributes and behaviour. A model type describes a concrete Modelling method, e.g. a business modelling method. Linking and relating the model types, forms a set of interrelated Modelling methods which describe a certain domain under consideration [3]. In this document the term Modelling method and Modelling language is used synonymously.

Modelling Methodology: A modelling methodology usually consists of a modelling procedure, a modelling method (language) and some modelling techniques.

Modelling Procedure: A modelling procedure describes the steps applying the modelling language to create results, i.e. models.

Modelling Technique: Modelling techniques are mechanisms and algorithms which allow the work on the models described by the modelling method (language).

Reference Model: A reference model is an abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details [4].

Workflow: A workflow is the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [5]. A workflow can also be defined as the orchestration of a set of activities to accomplish a larger and sophisticated goal. Examples of workflow include application processes, business processes, and infrastructure processes.

2. Process modelling notations and meta-models

Currently many standards are emerging in the scope of Process modeling. Some of these are a natural evolution of old specifications while others are brand new ones in this domain. The following table show the relevant standards organizations and the involved standards and its relationship:

Organization	Standards
Object Management Group - OMG - (www.omg.org)	Business Process Modeling Notation (BPMN) Business Process Definition Metamodel (BPDM) SPEM 2.0
Workflow Management Coalition - WfMC - (www.wfmc.org)	XML Process Definition Language (XPDL) Workflow API (WAPI) Workflow XML (WfXML)
OASIS (www.oasis-open.org)	Business Process Execution Language (BPEL)
W3C (www.w3c.org)	Open, collaborative review process SOAP, WSDL, core XML specifications Web Services Choreography Description Language (WS-CDL)
WS-I (www.ws-i.org)	Interoperability of WS technologies and standards WS-I Basic Profile

Table 1: Standards Organizations

In this section the most relevant Process Modeling Notations and Metamodels: SPEM, OPEN and BPDM will be analyzed, leaving the notations and business process languages standards focused for business processes for a later section.

2.1 SPEM 2.0

SPEM 2.0 (Software & Systems Process Engineering Metamodel) is a standard meta-model developed by Object Management Group (OMG)[6] whose main objective is to provide a formal framework for the definition of development processes of systems and software and the definition of the description on all the elements that make up. The specification has three versions released; the first one, called SPEM 1.0, was released in 2002. In 2005, SPEM 1.1 was released with minor updates but with several shortcomings as lack of enactment support and ambiguous semantics. SPEM 2.0 was released in 2007 fixing the before mentioned flaws and adding new features: compliant with UML 2, defining a new SPEM xml Schema compatible with MOF 2.0, alignment with emerging standards as BPMN and enabling the development of extensions for SPEM (used by tools automate processes).

The main objective of the SPEM 2.0 metamodel is to be able to maintain and support a wide range of fragments of method and processes of different styles, backgrounds, levels of formality, life cycle models, and communities to develop projects. SPEM 2.0 is agnostic to the development methodology that is used, making it possible to use in a variety of domains. For that the Software Process Engineering Meta-model SPEM defines a formal language for describing development processes. SPEM describes structures needed to formally express and maintain method content and processes, i.e. it defines a language and representation schema for method contents and processes. SPEM is not, however, intended to be a generic modeling language; rather, it defines the ability to choose the behavior modeling approach that fits the implementer's needs. SPEM uses the UML 2.0 infrastructure and diagram interchange standards [7].

A meta-modeling language (meta-meta-model) is used to describe the SPEM meta-model itself. The MOF 2.0 standard [8] provides such a language. As can be seen in the figure 1, the layers can be depicted as levels M0, M1, M2 and M3. M3 provides MOF which is instantiated by SPEM 2.0 on the M2 layer, in the same way that UML 2 meta-model instantiates MOF. On layer M1 there are concrete instances, such as 'Use Case', 'Analyst'. Layer M0 consists of the performing process.

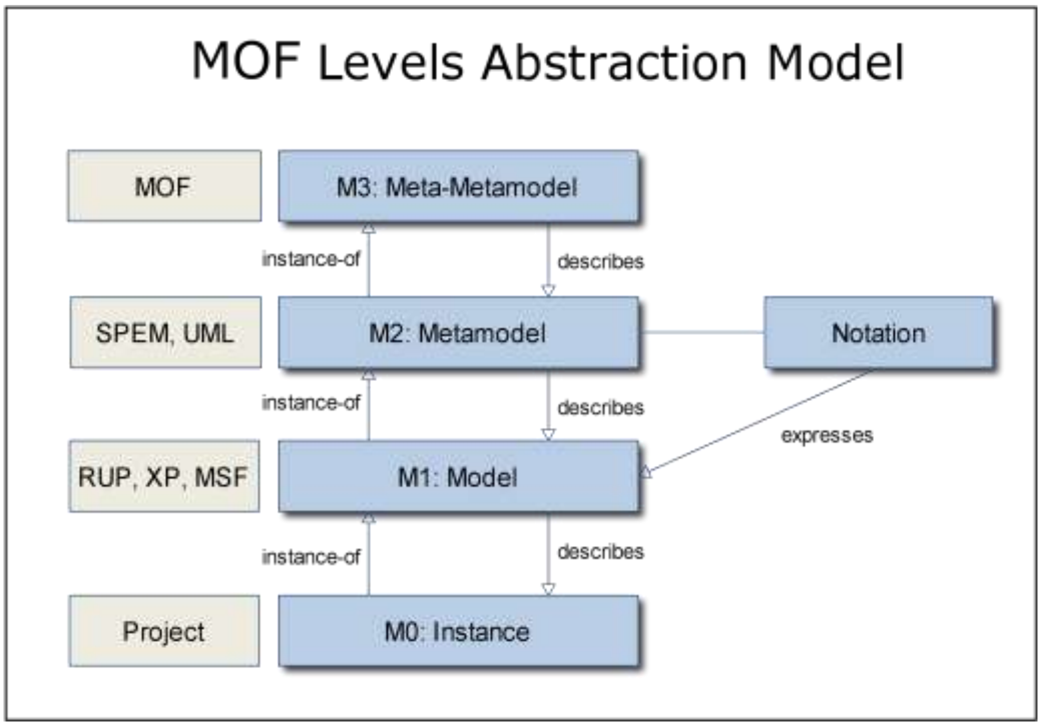


Figure 1: MOF levels of abstraction

SPEM defines the elements used in describing a process as well as elements used for structuring and managing this information.

The concept of SPEM to represent processes is based on three basic elements: roles, tasks and work products.

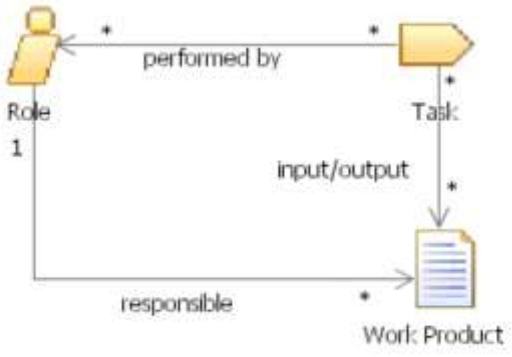


Figure 2: SPEM2.0 Basic Process

SPEM makes a clear separation of concerns between Method Content and Process. Method Content defines highly re-useable content. Process re-uses this content to create end-to-end processes or re-usable process "components".

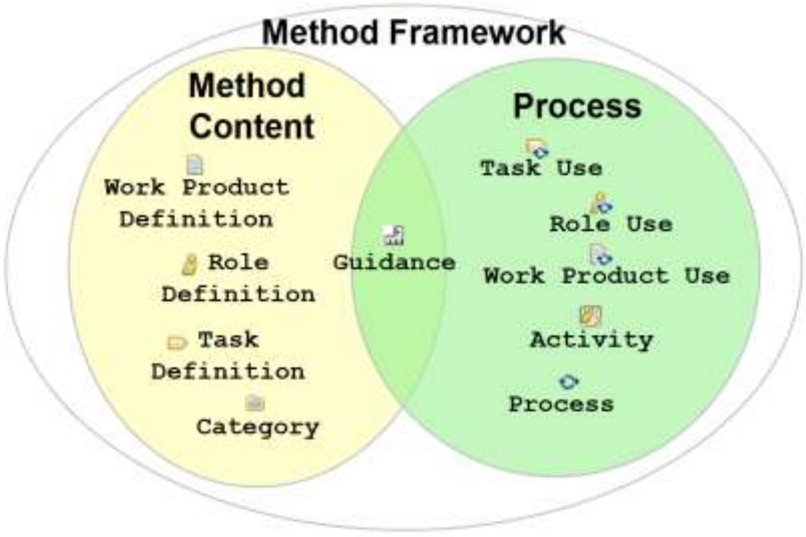


Figure 3: SPEM 2.0 Definition and Implementation

On one hand, the Method Content does not define when roles perform the tasks, but rather the relationships between roles, tasks, work products and associated guidance for these. The Method content indicate Who, What, Why and How without timing information and offering highly re-useable information.
On the other hand, Method Processes define the timing of tasks indicating When task are performed via Activity Diagrams and/or Work Breakdown Structures (and predecessor/successor relationships). It is therefore in charge for defining the development lifecycle through End-End sequence of Phases, Iterations, Activities and Milestones.

In the intersection of Content and Process are Guidance, which are templates, samples, roadmaps, etc. which give support to both methods and processes.

SPEM 2.0 has been designed for the development of the processes and can support different life cycle models offering flexible mechanisms to get variables and extensible processes.

A key feature of SPEM 2.0 is that define Methods Plug-in, which allow to customize the methods without changing its original content. As we will see after, one package that composes the metamodel SPEM 2.0, the Method Plug-In provides the capabilities to manage libraries of Method Content and Process allowing mechanisms of variability and extensibility through the reuse of Methods Content and Process certain. The types of variability are:

- **Method variability**: Mechanism that allows to customize method content without directly modifying the original. The concept is similar to inheritance in OO programming, and permits re-use with specialization. Content variability could be useful, for example, to change the description of an existing role, to add steps to an existing task, to add guidance to an existing task, and so on.

  There are four types of method variability:

  - **Contribute:** The contributing element adds content to the base element. Resulting published element is the base element + contributing element.
  - **Extends:** The contributing element inherits the content of the base element and specialized some or all of it. Both the base element and the extending element are published.
  - **Replace:** The replacing element replaces the base element. The resulting published element is the replacing element.
  - **Extends-Replace:** Similar to extends, however the base element is not published.

- **Process Variability**: Similar re-use mechanisms are available for Process content as well. In addition, Activities may also be created in the following ways:

  - **Extends:** The activity inherits the properties of the capability pattern. Updates to the capability pattern are automatically reflected in the activity.
  - **Copy:** An activity is created based on the capability pattern. It is not synchronized with the capability pattern.
  - **Deep Copy:** Similar to copy, but applied recursively to activities.

  - **Local Variability:** When a capability pattern is defined (either by Extends or Copy) local variability may be done (ex. Suppress steps in a task descriptor, change performing role, etc.).

Another remarkable feature is Patterns of reusable processes and best practices for assembling rapid processes. The patterns of processes defined in SPEM 2.0, are blocks used to create new development processes; two main patterns of processes exist:

- **Capability patterns**: Define the sequence of work (i.e. the sequence of tasks) to achieve a particular purpose. They are similar to the work breakdown structures (WBS) used in project management tools and define the timing of tasks, including any predecessor/successor relationships. The elements of the WBS can be specialized within the context of a Capability Pattern. In the same way Capability patterns may be nested, to build up larger building blocks.
- **Delivery Process:** A delivery process defines an end-to-end specification for achieving a goal (ex. release a new version of a software application) defining end-end full-lifecycle process. It can be seen as the top-level activity. In addition to activities, delivery processes may contain milestones, phases, and iterations which are defined using Work Breakdown Structures and/or Activity Diagrams.

2.1.1 Metamodel structure

SPEM 2.0 presents a metamodel structured in seven main packages:

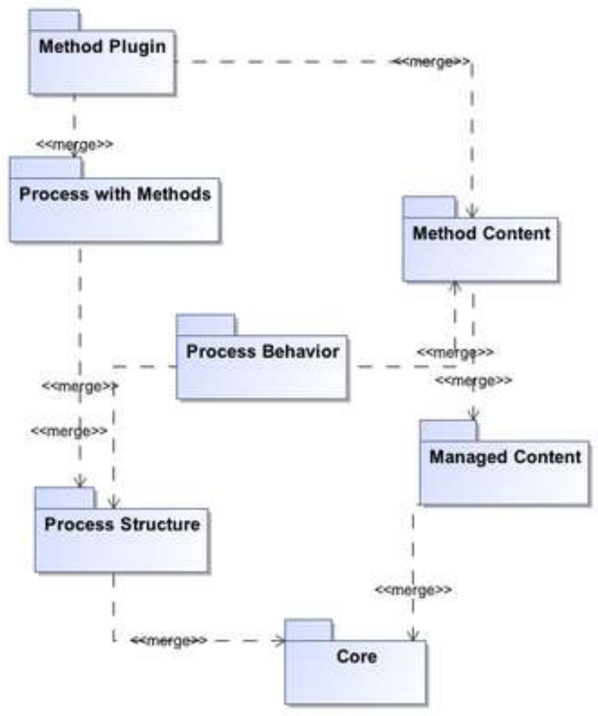


Figure 4: SPEM 2.0 Metamodel Structure

This structure divides the metamodel in logical units. Each of these units complements and extends the units on which it depends, providing it with additional structures. Each of the units defined in the lower layers of the structure can be understood as an implementation of SPEM 2.0 without using the higher level units. In many cases, the classes in the metamodel are defined in a simple lower-level units and are then extended to larger units through the mechanism of combining packages with additional properties and relationships to meet more complex requirements of process modeling.

The packages provide the following capabilities, described in more detail in the following subsections:

Core: The package Core of Metamodel SPEM 2.0 contains all classes and abstractions that constitute the basis for the rest of the packages of metamodel.

Process Structure: Contains the necessary classes for creating process models. Supports creation of a simple and flexible process models.

Process Behavior: Can represent the dynamic part of the processes, the behavior.

Managed Content: This package will allow to provide processes or systems with annotations and descriptions that can be expressed as models and therefore should be documented and managed as natural language descriptions.

Method Content: Contains all the elements for creating a set of reusable methods, its aim is to illustrate which are the goals that a method has to reach, which resources are used and which roles are involved.

Process With Methods: Composed of the main elements for modelling a process: Activities, nested in a breakdown structure where the performing Role classes and the input and output Work Product classes for each activity are listed.

Method Plug-In: This package introduces the concepts to design, manage and maintain repositories and libraries of Method Content and Process.

2.1.1.1 SPEM Core

The Core Package is the basis on which the rest of SPEM2 packages are built. The main classes in this package support the two main capabilities of SPEM: creating classifications of SPEM2 classes depending on needs and having available a set of abstract classes to describe the work through SPEM 2 Process.

The most relevant classes of this package are:

- **WorkDefinition**: It is an abstract class which allows generalizing any kind of work inside of a SPEM 2 specification. Also is one of the most important elements of SPEM specification.
- **WorkDefinitionParameter**: It is a generalization of the Process elements representing parameters for input or output for a specific Work Definition.
- **WorkDefinitionPerformerMap**: It is an abstract class which allows to generalize the relation between the Work Definition and the responsible to perform a job.

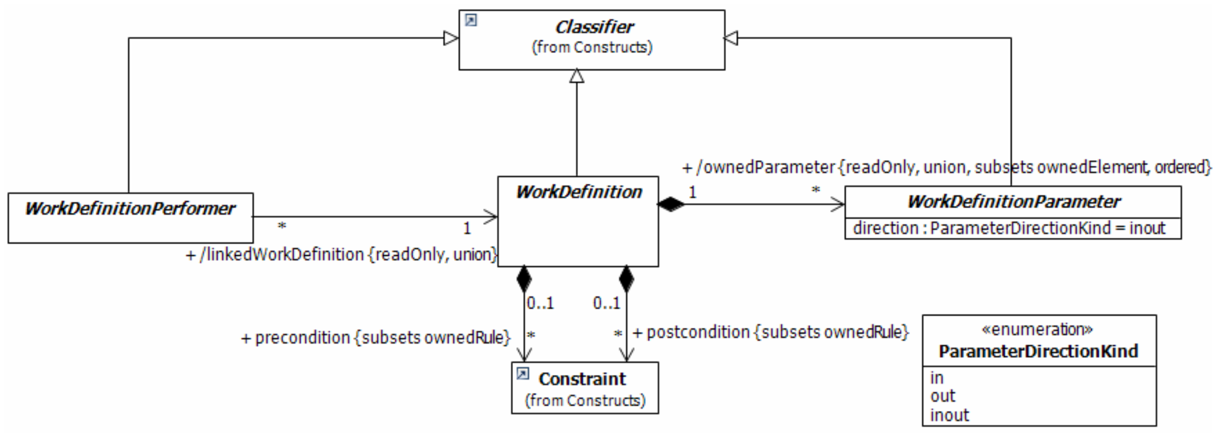

Classifier
(from Constructs)
WorkDefinitionPerformer
WorkDefinition
WorkDefinitionParameter
direction : ParameterDirectionKind = inout
+ /ownedParameter {readOnly, union, subsets ownedElement, ordered}
1
*
+ /linkedWorkDefinition {readOnly, union}
0..1
+ precondition {subsets ownedRule}
+ postcondition {subsets ownedRule}
Constraint
(from Constructs)
«enumeration»
ParameterDirectionKind
in
out
inout

Figure 5: SPEM Core Diagram

2.1.1.2 SPEM Process Structure

The Process Structure package contains the basic elements to define development process through a mechanism of decomposition. The main element is the activity (Activity) which in turn can be decomposed into other activities and contain other types of items such as Milestones, Role Uses etc.

The most relevant classes of this package are:

- **Activity**: It is the element used to represent the basic unit of work or to represent a process itself.
- **Milestone**: A significant element in the development process.

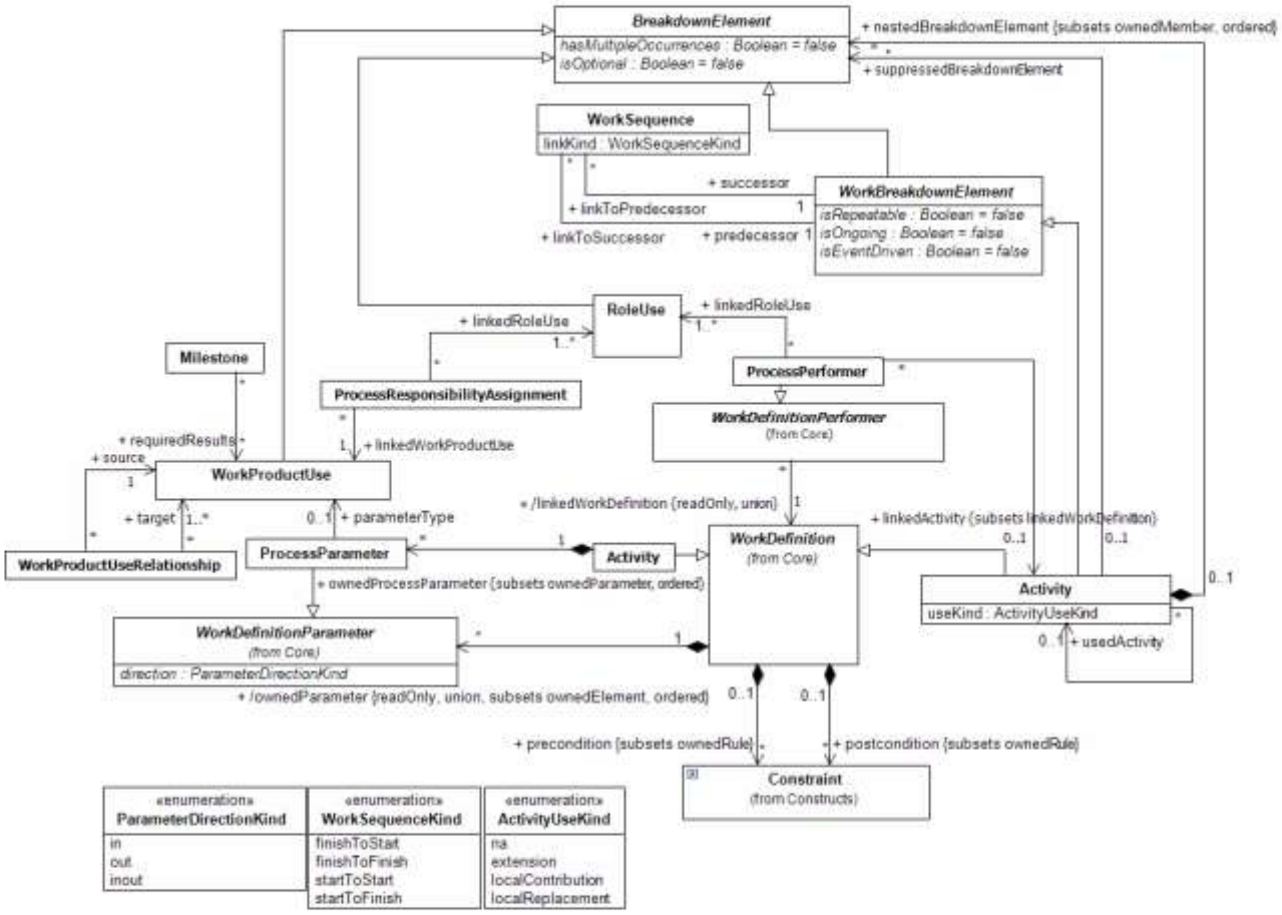

BreakdownElement
hasMultipleOccurrences : Boolean = false
isOptional : Boolean = false
+ nestedBreakdownElement {subsets ownedMember, ordered}
+ suppressedBreakdownElement
WorkSequence
linkKind : WorkSequenceKind
+ successor
+ linkToPredecessor
+ linkToSuccessor
+ predecessor
WorkBreakdownElement
isRepeatable : Boolean = false
isOngoing : Boolean = false
isEventDriven : Boolean = false
RoleUse
+ linkedRoleUse
+ linkedRoleUse
Milestone
ProcessResponsibilityAssignment
ProcessPerformer
WorkDefinitionPerformer
(from Core)
+ requiredResults
+ source
+ linkedWorkProductUse
WorkProductUse
+ target
+ parameterType
+ /linkedWorkDefinition {readOnly, union}
+ linkedActivity {subsets linkedWorkDefinition}
WorkProductUseRelationship
ProcessParameter
Activity
WorkDefinition
(from Core)
Activity
useKind : ActivityUseKind
+ usedActivity
+ ownedProcessParameter {subsets ownedParameter, ordered}
WorkDefinitionParameter
(from Core)
direction : ParameterDirectionKind
+ /ownedParameter {readOnly, union, subsets ownedElement, ordered}
+ precondition {subsets ownedRule}
+ postcondition {subsets ownedRule}
Constraint
(from Constructs)
«enumeration»
ParameterDirectionKind
in
out
inout
«enumeration»
WorkSequenceKind
finishToStart
finishToFinish
startToStart
startToFinish
«enumeration»
ActivityUseKind
na
extension
localContribution
localReplacement

Figure 6: SPEM Process Structure Diagram

2.1.1.3 SPEM Process Behaviour

SPEM 2.0 does not impose any notation for modeling the dynamic behavior of the system, just defines how link the models created with other supported notations languages to reflect the behavior as BPMN, Activity Diagrams, etc.. With the elements of the package Process Structure.

2.1.1.4 SPEM Managed Content

The Managed Content package defines the fundamental concepts for managing textual descriptions for process and method content elements. The main element is the abstract class Describable Element which is the superclass of all elements of the system which can have a textual description.

The most relevant classes of this package are:

- **Category**: Categories can be used to categorize content based on the user's criteria as well as to define whole tree-structures of nested categories allowing the user to systematically navigate and browse method content and processes based on these categories. Normally used to group related method elements.

  Categories may be nested and there are five Standard predefined Categories, but open to define own custom categories; Predefined categories are:
  - **Discipline**: grouping of related tasks.
  - **Domain**: grouping of related Work Products.
  - **Work Product Kind**: similar to Domain.
  - **Role Set:** Grouping of related Roles.
  - **Tool**: Grouping of Tools.
- **Describable Element:** Store textual descriptions for a Describable Element.
- **Guidance**: Element that provides additional information about any Describable Element, for example how use a tool or how to perform a task. If Tasks should indicate "what" needs to be done, Guidelines provide detailed "how to". Then Guidance may be associate with Roles, Tasks, and Work Products and depending upon purpose they have different types.

  Can be of different types as Checklist, Templates, etc.. the following table presents the different types of Guidance:

Guidance	Description
Checklist	Identifies a series of items that need to be completed or verified. Checklists are often used in reviews such as walkthroughs or inspections.
Concept	Outlines key ideas associated with basic principles underlying the referenced item. Concepts normally address more general topics than guidelines and span several work product and/or tasks or activities.
Example	Provides an example of a completed work product or performed tasks and activities.

Estimate	Provides sizing measures, or standards for sizing the work effort associated with performing a particular piec4e of work.
Estimation Considerations	Indications for estimating the effort associated with some work, including considerations on how to estimate and the metrics used
Estimating Metric	Describes a metric or measure that is associated with an element and which is used to calculate the size of the work effort as well as a range of potential labor
Guideline	Provides additional detail on how to perform a particular task or grouping of tasks, or that provides additional detail, rules, and recommendations on work products and their properties.
Practice	Represents a proven way or strategy of doing work to achieve a goal that has a positive impact on work product or process quality. Practices are defined orthogonal to methods and processes. They could summarize aspects that impact many different parts of a method or specific process.
Report	A template for an automatically generated description with extracted content one or several work products.
Reusable Asset	Links a prepackaged solution to a problem for a given context. Examples of asset are design patterns or mechanisms, solution frameworks, etc.
Roadmap	A linear walkthrough of a complex process or activity. (This is process specific guidance.)
Supporting Material	A catch-all for other types of guidance not specifically defined elsewhere. It can be related to all kinds of content elements.
Template	For a work product, provides a predefined table of contents, sections, packages, and/or headings, a standardized format, as well as descriptions on how the sections and packages are supposed to be used and completed.

Term Definition	Defines terminology and is used to build up the Glossary.
Tool Mentor	Shows how to use a specific tool to accomplish some piece of work, either in the context of, or independent from, a task or activity.
Whitepaper	Similar to concept, but has been externally reviewed or published and can be read and understood in isolation of other content elements and guidance.

Table 2: Types of Guidance

- **Section:** Structure elements descriptions in different parts.
- **Metric:** Defines measures on Describable Elements.

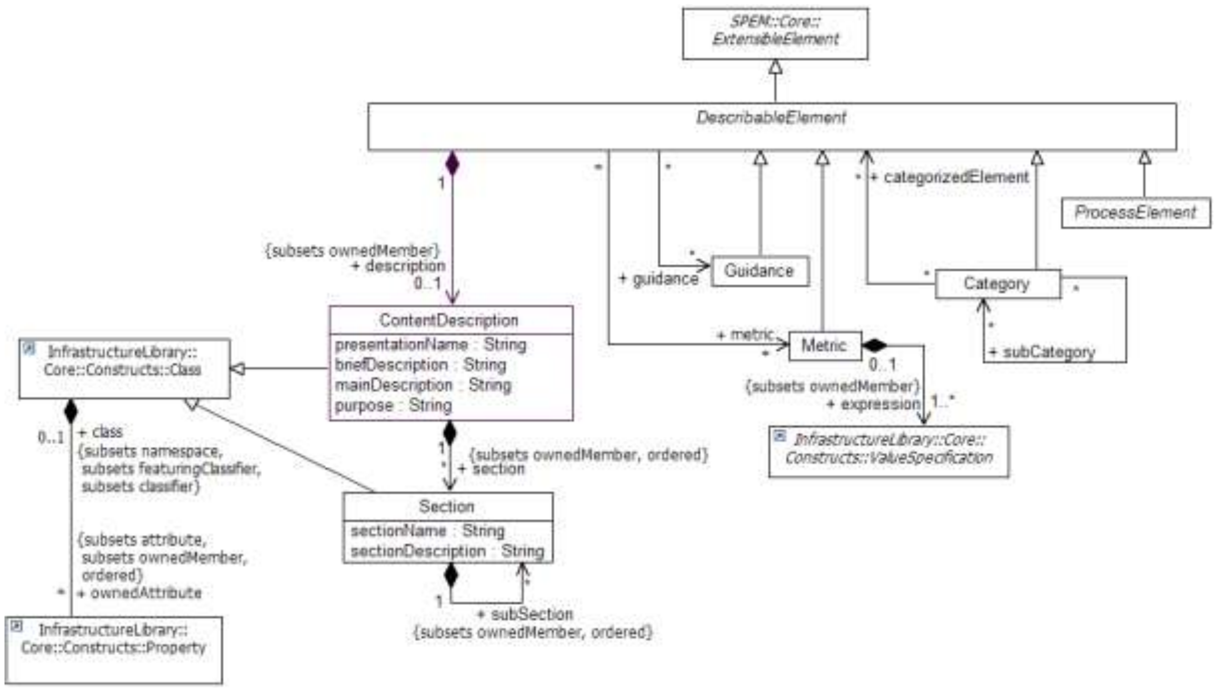

SPEM::Core::ExtensibleElement
DescribableElement
ProcessElement
+ categorizedElement
Guidance
+ guidance
Category
+ subCategory
Metric
+ metric
0..1
{subsets ownedMember}
+ expression
1..*
InfrastructureLibrary::Core::Constructs::ValueSpecification
{subsets ownedMember}
+ description
0..1
ContentDescription
presentationName : String
briefDescription : String
mainDescription : String
purpose : String
InfrastructureLibrary::Core::Constructs::Class
+ class
{subsets namespace, subsets featuringClassifier, subsets classifier}
{subsets attribute, subsets ownedMember, ordered}
+ ownedAttribute
InfrastructureLibrary::Core::Constructs::Property
{subsets ownedMember, ordered}
+ section
Section
sectionName : String
sectionDescription : String
+ subSection
{subsets ownedMember, ordered}

Figure 7: SPEM Managed Content Diagram

2.1.1.5 SPEM Method Content

The Method Content package contains the main elements of the fragments of methods such as roles, tasks, steps and WorkProduct Definitions. The main purpose of this package is to define the set tasks (Task Definition), organize them into different steps (Steps), define what each (Work Product Definition) i/o produces and specify who has made this task (Role Definition).

Relevant classes/elements in this package are:

- **Task Definition**: Defines an assignable unit of work, and are performed by Roles. Also have a clear purpose, and provide step-by-step descriptions of the work that needs to be done to achieve an specific goal. In these steps Tasks modify or produce Work Products but do not define "when" they are performed in the lifecycle. This description is completely independent of when in a process lifecycle the work would actually be done. It therefore does not describe when you do what work, but describes all the work that gets done throughout the development lifecycle that contributes to the achievement of this goal. When the Task Definition instance is applied in a process, then this process application provides the information of which pieces of the Task Definition will actually be performed at any particular point in time.
- **Role Definition**: Defines a set of skills, competencies and responsibilities of a participant or a set of participants. Roles are not individuals or resources. Individual members of the development organization will perform different roles.
- **Work Product Definition**: Defines anything that is consumed, produced or modified by the tasks. They may serve as a basis for defining reusable assets. Roles use Work Products to perform Tasks and produce Work Products in the course of performing Tasks.

  Work Products are the responsibility of Role Definitions, making responsibility easy to identify and understand, and promoting the idea that every piece of information produced in the method requires the appropriate set of skills.

  There are three types of Work Products:

  - **Artifact**: typically a configuration managed item. An artefact could consist by other simpler artifacts.
  - **Deliverable**: required customer/stakeholder deliverable.
  - **Outcome**: intangible result of a task such as an installed server or tool.
- **Step**: Used to organize a task in parts or subunits of work.

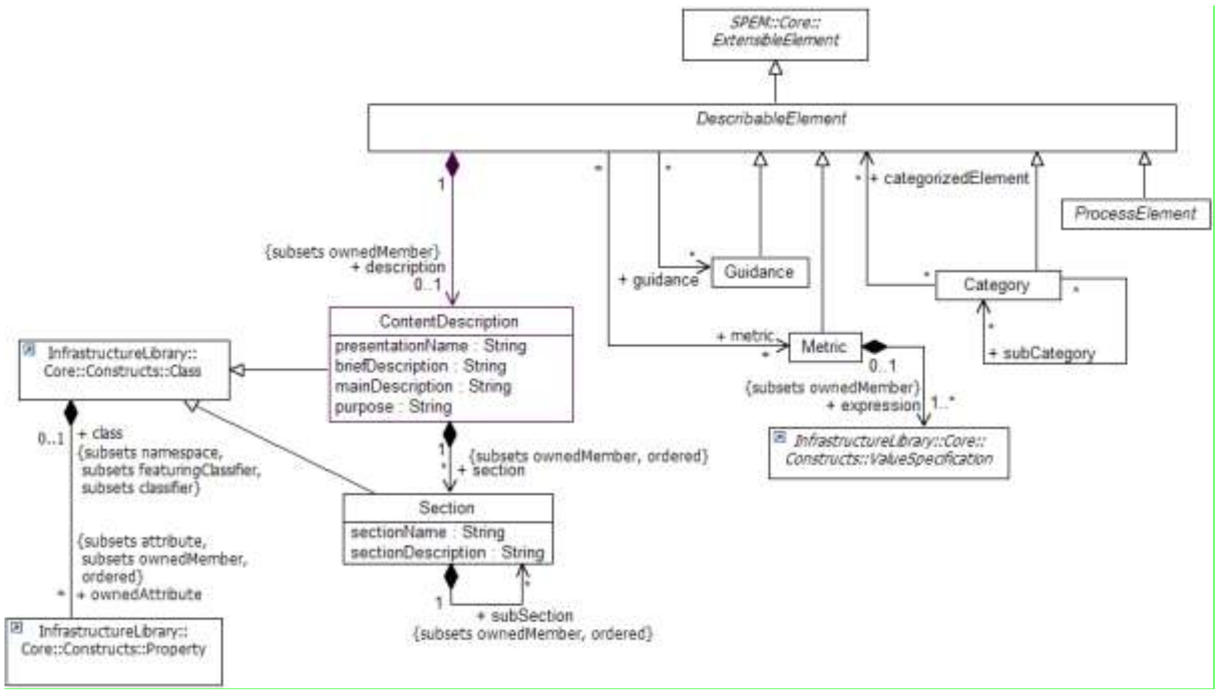

SPEM::Core::ExtensibleElement
DescribableElement
ProcessElement
Guidance
Category
Metric
ContentDescription
presentationName : String
briefDescription : String
mainDescription : String
purpose : String
Section
sectionName : String
sectionDescription : String
InfrastructureLibrary::Core::Constructs::Class
InfrastructureLibrary::Core::Constructs::Property
InfrastructureLibrary::Core::Constructs::ValueSpecification
{subsets ownedMember} + description
+ guidance
+ categorizedElement
+ subCategory
+ metric
{subsets ownedMember} + expression
{subsets ownedMember, ordered} + section
+ subSection {subsets ownedMember, ordered}
+ class {subsets namespace, subsets featuringClassifier, subsets classifier}
{subsets attribute, subsets ownedMember, ordered} + ownedAttribute

Figure 8: SPEM Method Content Diagram

2.1.1.6 SPEM Process With Methods

The Package Process With Methods provides the data structures necessary to reflect good practices in the industry and allow reuse between separate instances of processes and methods.

Relevant classes/elements in this package are:

- **Activity**: Is defined as a set of different elements (activities, task uses, roles, milestone... etc.) defined in a namespace, and for which has been described a series of relationships according to the method or project that we are.
- **Method Content Use**: it is the key concept to understand the separation among Process and Method Content. The Content Method has a series of elements and relationships that are modified to a particular Process for which the Content Method has been created.
- **Process Package**: A package that can only contain Process Elements.
- **Role Use**: A role in the context of a particular activity.
- **Task Use**: Task Definition in the context of a particular activity.

- **Work Product Use**: Work Product definition within the context of a particular activity.

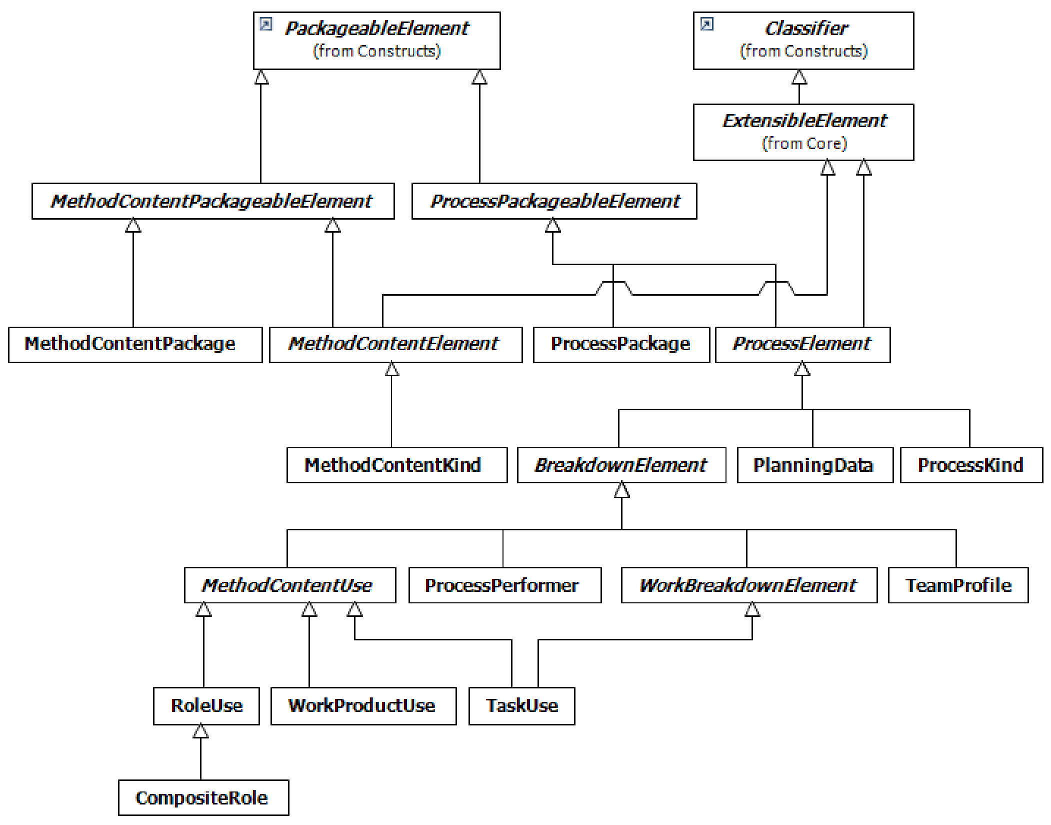


Figure 9: SPEM Process with Methods Diagram

2.1.1.7 SPEM Method Plug-In

The Package Method Plug-In provides capabilities to manage libraries of Method Content and Process allowing variability and extensibility mechanisms by reusing Method Content and certain Process. A Method Plug-in is a container of fragments of method and is formed by Method Content and Processes. The way to create a Method Plug-in, SPEM 2.0 allows referencing other Plug-ins whose content want reuse and/or extend.

Relevant classes/elements of this package to structure and manage the information are:

- **Method Plugin**: Is defined as a storage unit of the configuration, modularization, packaging and deployment of Process and Method Content. It can be seen as a container for content that can be independently imported and exported from a library. This kind of element can re-use information in other plug-ins. Method Plug-ins are further sub-divided into Method packages and Process package to simplify managing the information.

- **Method Library**: Is defined as a container of Method Plugins and definitions of Method Configuration.
- **Method Configuration**: Is a logical subset inside of Method Library defined using a selection of Methods Packages. It can be seen as a 'filter' applied to the library that gives only what is needed for a particular purpose.
- **Variability Element**: Used to provide capabilities for variation and extension in the SPEM elements.

2.1.2 SPEM 2.0 as UML Profile

In addition of metamodel, SPEM 2 is defined as a UML profile; this makes it easy to work with existing UML tools. SPEM 2 and UML are located in the level of abstraction M2 of the OMG (see figure 1).

Stereotype	Meta-/Superclass	Icon
Activity	WorkDefinition, Planned Element / Action	👥
Category	DescribableElement / Class	📁
CompositeRole	RoleUse	👥
Guidance	DescribableElement / Class	📐
Process	Activity	🔄
RoleDefinition	MethodContent Element / Class	👤
RoleUse	BreakdownElement / Classifier	👤
TaskDefinition	MethodContentElement, WorkDefinition	▭

TaskUse	WorkBreakdownElement, PlannedElement / Classifier, Action	📂
WorkProductDefinition	MethodContent Element / Class	📄
WorkProductUse	BreakdownElement / Classifier	📄

Table 3: Relevant SPEM 2 UML profile Stereotypes

2.1.3 Tools

Both Open Source and Commercial tools are present in the market, allowing descriptions using the SPEM2.0 recommendations:

- **EPF Composer**

  Eclipse Process Framework Composer is an SPEM 2.0 editor is an Open Source tool result of the Eclipse Process Framework (EPF) project. The EPF Composer uses a forms-based approach to defining method content (such as roles, tasks, and work products) in an Eclipse IDE. Method content is then configured into process patterns using various breakdown structures and activity models.

  Source:
http://www.eclipse.org/projects/project_summary.php?projectid=technology.epf

- **IBM Rational Method Composer (RMC)**

  The commercial version of the EPF Composer

  Source: http://www.ibm.com/software/awdtools/rmc

- **PRO3**

  Commercial tool developep on top of EPF and Microsoft Project developed by Objecteering.

  Source: http://www.Objecteering.com

2.1.4 Conclusions

SPEM 2.0 establishes a clear separation between the formal definition of a method (Method Content) and its possible use within a specific process in a concrete project.

At the same time it allows to easily define new processes assembling existing parts, located in repositories (store and manage processes for use in later developments). Also recognizes the need that might exist certain parts that cannot be formalized and therefore they should be included within the process through a description in natural language.

The SPEM 2.0 Specification comes in the form of a MOF2.0 Compliant metamodel that reuses UML2.0 Infrastructure and UML 2.0 Diagrams interchange specifications and also comes as UML Profile where each element of the SPEM 2.0 Specification is defined as a stereotype in UML2.0. This allows the use of the multiple UML tools existing in the market and aims to facilitate the vendors adoption to this specification.

Finally it leaves open to developers the selection of the notation to describe the dynamic behavior, making it easy to select the best suited notation to the domain of the problem. Nevertheless, it suggests to use UML2.0 Activity Diagrams or BPMN as the they support the most workflows patterns.

2.2 OPEN

2.2.1 General overview

Object-oriented Process, Environment, and Notation (OPEN) is a free, public domain, defacto industry-standard approach for the production of endeavor-specific development methods. OPEN was originally created in the mid-1990s as a merger of several earlier object-oriented software development methods. Since then, OPEN has grown and evolved to support the development, sustainment, and retirement of software-intensive systems as well as to support business [re]engineering.

Object-oriented Process, Environment, and Notation (OPEN) is the premier third-generation, public domain, full lifecycle, process-focussed, methodological approach that was designed for the development of software intensive applications, particularly object-oriented and component-based developments. OPEN was developed and is maintained by the not-for-profit OPEN Consortium.

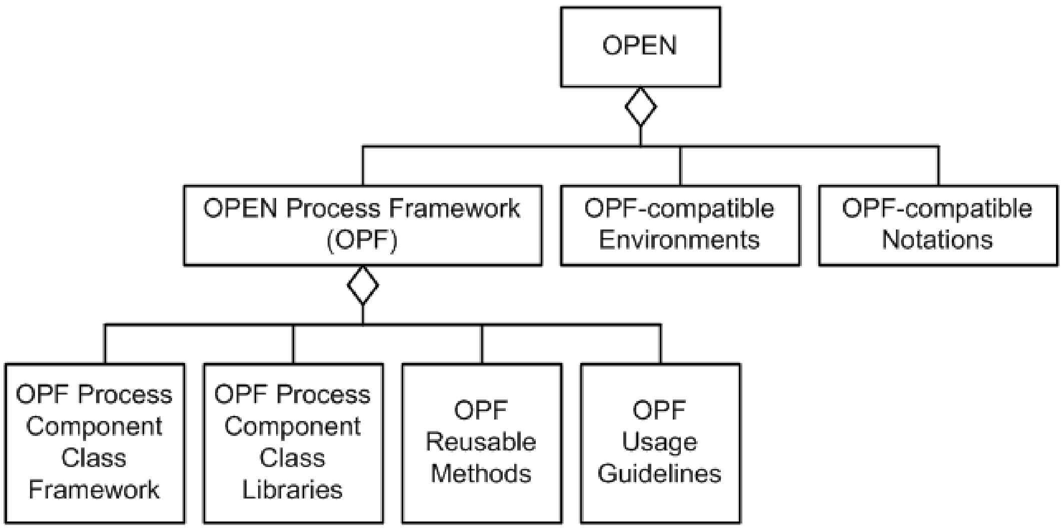

OPEN
OPEN Process Framework (OPF)
OPF-compatible Environments
OPF-compatible Notations
OPF Process Component Class Framework
OPF Process Component Class Libraries
OPF Reusable Methods
OPF Usage Guidelines

Figure 10: the OPEN Model

As illustrated in the preceding figure, OPEN is composed of the following three parts:

- **OPEN Process Framework (OPF)**, which is a framework for process engineering (method engineering) consisting of the:
  - OPF Process Component Class Framework, which is a process metamodel that defines the standardized core abstract process component classes (method components), the relationships between them, and any constraints (i.e., well-formedness rules) that constrain these classes and their relationships.
  - OPF Process Component Class Libraries, which consists of free, open-source, reusable process component subclasses of the classes in the process metamodel.
  - OPF Reusable Methods, which are reusable process models made by integrating appropriate concrete process component classes from the OPF class libraries.
  - OPF Usage Guidelines, which are guidelines that are used to construct new OPF-compatible process models (a.k.a., methods) or extend and tailor ones.
- **OPF-compatible Environments**, which are integrated sets of one or more tools that can be used for process engineering with the OPF.

- **OPF-compatible Notations**, which are one or more notations (e.g., OML, UML) that are appropriate for documenting the classes of process components stored in the OPF repository.

The OPEN metamodel defines five main, high-level classes of types of method fragments:

- **Work Product**: anything of value that is produced, used, or modified during the process of development. Work Products are the result of producers (usually people) executing Work Units and are used either as input to other Work Units or delivered to a client. Despite the name (viz. "products"), they also include externally supplied (e.g., by a user) pre-existing artefacts used as inputs to Work Units.

- **Producer**: entity responsible for creating, evaluating, iterating, and maintaining Work Products—typically a person, but could also be a software tool or even a piece of hardware.

- **Work Unit**: a functionally cohesive operation that is performed by a Producer. There are three major classes of Work Unit: Activity, Task, and Technique. Activities and Tasks are statements at different levels of granularity of what needs to be done. Techniques, on the other hand, delineate the how. Because of their visible and tangible nature, Work Products, and especially those that are deliverable, are of particular interest to project managers. They also provide an alternative to Tasks in project tracking.

- **Language**: a medium for documenting a Work Product, for example, natural language, UML, Java.

- **Stage**: an identified and managed duration within the process or a point in time at which some achievement is recognized. Stages describe when things happen in the software development lifecycle.

Activities in OPEN are coarse, granular descriptions of what needs to be done. The combination of Activity objects with forward and backward links form the process. Scheduling comes from the planning Activities and Tasks combined with the project management elements embodied in the appropriate Tasks. A specific process instantiation may concentrate on a single project or on a programme of several projects, which introduces new foci such as domain modelling and reuse, as well as resource allocation.

Also focusing on the "what" is the OPEN Task, which offers much more detail than the OPEN Activity and provides project management support. While OPEN's Tasks are like Activities in the sense that they describe jobs to be done (but not how to do them), they are different in that Activities are conceptual, while Tasks, as the smallest unit of work that can be project-managed and result in a deliverable, are linked to a project management mindset.

In addition, the idea that business culture should be adapted to fit a specific methodology is not good business sense, despite its prevalence in many of the marketed methodologies to date. When using IT and its dependent parts, such as methodologies, it is critical that they fit the business and not the other way around.

2.2.2 Tools

Support for the OPEN process currently is relative poor, and very discontinuous. The tools that implement this standard are:

- **eTrack**

  Commercial tool developep by eTrack Products Pty Ltd..

  Source: www.etrack.com.au

- **ArcStyler**

  Commercial tool developep by Interactive Objects Software GmbH.

  Source: www.arcstyler.com

- **Process Continuum**

  Commercial tool developed by Myriad Solutions.

  Source: www.myriadsolutionsinc.com

2.2.3 Conclusions

OPEN is fully object-oriented Methodology/process of public domain, developed a few years ago which supports business process modeling among others and provides strong support for the full lifecycle of the software application;

Key features of OPEN are threefold:

- Software quality
- The use of metrics and
- The reuse of processes

OPEN encapsulates business, quality, modelling and reuse issues within its end-to-end lifecycle support for software development using the object-oriented paradigm.

OPEN is very flexible and permits to be used on either small projects or large, and independently of the criticality of the project mission. The flexibility is applicable in either short lifetime projects or on long-term business core software that needs to ensure the perdurability and quality of it.

The metamodel-based framework of OPEN can be tailored to individual domains or projects taking into account personal skills, organizational culture and requirements peculiar to each industry domain.

Furthermore seems that other standards such as SPEM are pushing over and are presented as substitutes, the reason for that might be the fact that permits more expressivity and has better tooling support.

2.3 BPDM

The Business Process Definition Metamodel (BPDM) [9] is a standard XML-based proposal for business processes being developed by the Object Management Group (OMG). The final submission of the proposal of BPDM is dated as November 2008.

The BPDM is a framework which supports the representation of business processes independent from notation or methodology and provides an explicit MOF based metamodel and a robust serialization mechanism based in XMI, which is going to allow exchange business process specifications between modeling tools, and between tools and execution environments.

The main goals to achieve BPDM are:

- Obtain a Common metamodel to unify the diverse business process definition notations that exist in the industry containing semantics compatible with leading business process modeling notations. Nevertheless BPDM provides a formal extension to BPMN, as the BPMN concepts are represented in an explicit metamodel. Thus it may support interoperability between different tools and notations.

- The ability to integrate process models for workflow management processes, automated business processes, and collaborations between business units.
- Support for the specification of web services choreography, describing the collaboration between participating entities and the ability to reconcile the choreography with supporting internal business processes. For that it defines a shared vocabulary for process modeling concepts distinguishing complementary views of a process:
    - Orchestration: Includes the traditional view where sequences of activities are carried out.
    - Choreography: Describes interactions of entities each of which may have their own internal orchestration processes.

The BPDM package contains the models for orchestration (including BPMN) and choreography, and their performance, enactment, and execution. It has six subpackages grouped into two categories:

- **Common Behavior Model**: for the aspects of dynamics in common between orchestrations and choreography (Behavior Model, and Interactive Behavior Model).
- **Activity Model**: for orchestration and Interaction Protocol Model for choreography. BPMN Extensions are including here. Is what the business process does.

The BPDM package imports the Common Infrastructure package which provides the framework that ties the other models to performance, enactment, and execution (Abstractions, Composition Model, Course Model and Condition Model).

BPDM uses the terms "Process" and "Interaction Protocol" instead of orchestration and choreography in order to avoid misinterpretation.

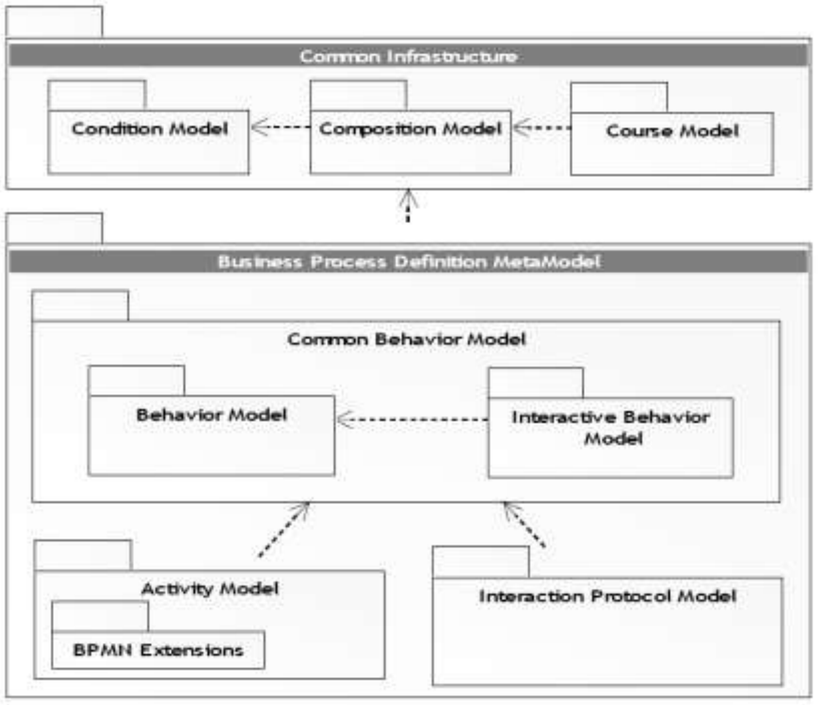


Figure 11: BPDM Metamodel Structure

BPDM is often compared to the existing process interchange format XPDL, but although the efforts are similar in that they could be used by process design tools to exchange business process definitions, the key difference is that BPDM is based on XMI, a format for exchange of programming models from the OMG. And the most important is that BPDM has not yet been ratified nor implemented at this time.

The model is an UML clarification of BPMN concepts. This gives the basis for a software development environment (e.g. IDE) to model workflows with UML and MOF based BPMN graphical interface, rather than using some proprietary graphical workflow design interface.

2.3.1 Conclusions

BPDM Provides abstract concepts to express business process models and its mission acts as an interchange channel among different business process description languages. At the same time presents a unification of orchestration and choreography and is designed to represent concepts from a business perspective.

It was initially designed to supplement BPMN with a formal metamodel of its modeling constructs and envisioned to become persistency format for BPMN. A good set of features such as Mapping to MOF and XMI are part of the BPDM. At the beginning BPMD specifies a mapping between BPDM and BPMN, but currently has been absorbed by BPMN 2.0 and is included in the latter one.

3. Business process modeling

3.1 Business Process Management (BPM)

Business Process Management (BPM) is ultimately concerned with the management of a process model that is a run-time executable artifact that drives the business, and is monitored for improvement. A process developed within a BPM context will follow a full lifecycle from analyse/design through testing, deployment, execution and measurement/monitoring.

For a customer the benefits of BPM can be quite far reaching. It provides a 'process layer', by removing process from business rules etc embedded in specific applications, and exposes the business processes in a way that enables the business to manage, improve and monitor performance without the need for IT.

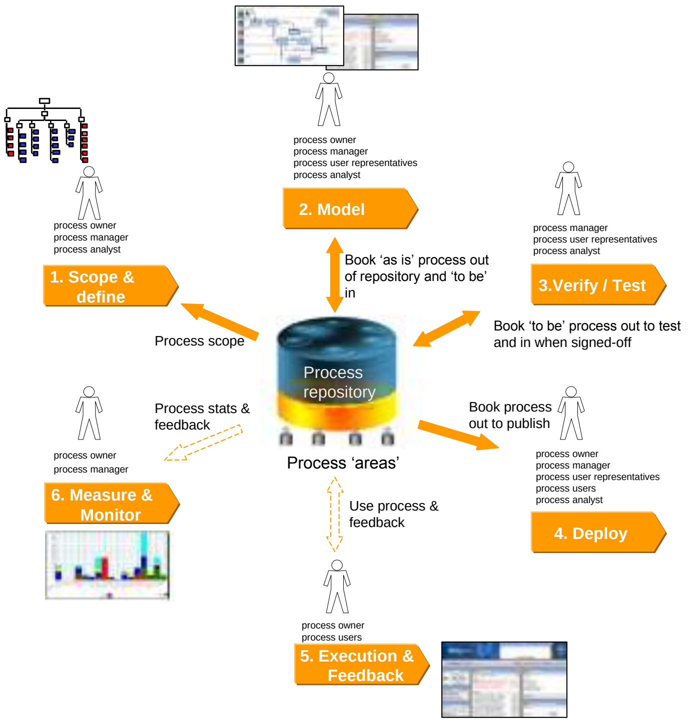


Figure 12: Sample BPM lifecycle

3.1.1 Methodology vs. Method

The main difference between modeling methodology and modeling language or method is that a modeling methodology is rather a vision of business process modeling with its intended use and steps to create models, while modeling language translates these sometimes abstract ideas into practice. Methodology is rather a way of thinking, while language is a set of tools that allow describing reality on the basis of the chosen paradigm.

A modelling methodology usually consists of a modelling procedure, a modelling language (method) and some modelling techniques. The modelling language contains the elements to describe a model and is characterized by its syntax, semantics and notation. A modelling procedure describes the steps applying the modelling language to create results, i.e. models. Modelling techniques are mechanisms and algorithms which allow the work on the models described by the modelling language.

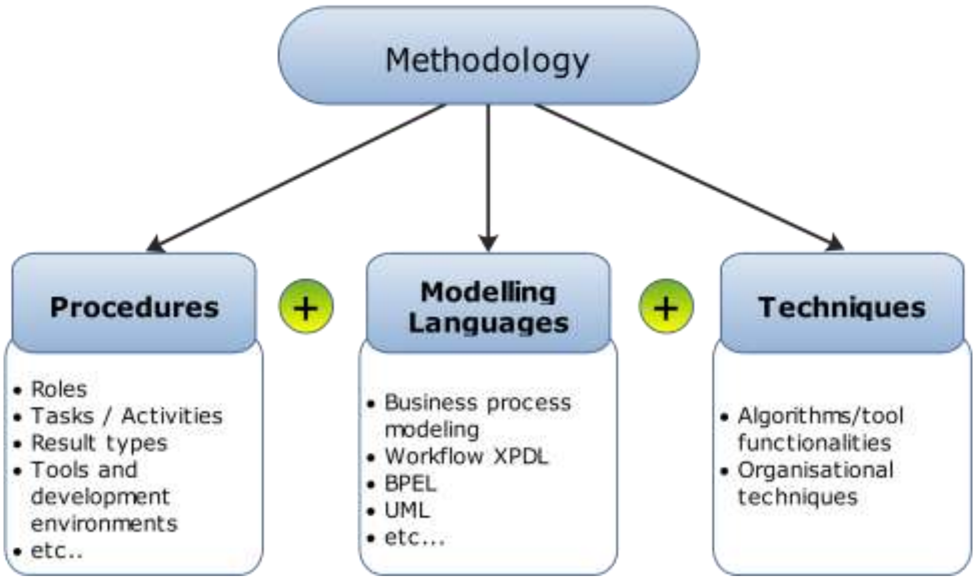


Figure 13: Methodology vs. Modelling language

3.2 Business Process Modelling Languages (BPML)

Business Process Modelling Languages (BPMLs) express certain aspects of processes (e.g. activities, roles, interactions, data, etc.) and address different application areas.

The most important by its adoption in research and industry fields, UML 2.0 activity diagrams, BPMN 2.0, EPC, IDEF and BPML are analyzed in this subchapter.

3.2.1 UML 2.0 Activity Diagrams

The Unified Modeling Language (UML) [10] is a standard modeling language currently maintained by the Object Management Group (OMG), for visualizing (using the standardized graphic UML notations) and specifying the static structure, dynamic behavior and model organization. UML consists of a notation for describing the syntax of the modeling language and a graphical notation and a meta-model which describes the static semantics of UML. The UML specification consists of the Infrastructure which defines foundational language constructs required for UML and of the Superstructure which defines user level constructs (diagrams) and it is defined by a Meta-Object Facility (MOF) metamodel.
According to the OMG's description, UML 2.0 defines thirteen different types of diagrams, divided into three categories: Six diagram types for modeling of system structures and details of the static system; three to model the dynamic behavior of a system; and four represent different aspects of interactions:

- **Structure Diagrams** include the Class Diagram, Object Diagram, Component Diagram, Composite Structure Diagram, Package Diagram, and Deployment Diagram.
- **Behavior Diagrams** include the Use Case Diagram (used by some methodologies during requirements gathering); Activity Diagram, and State Machine Diagram.
- **Interaction Diagrams**, all derived from the more general Behavior Diagram, include the Sequence Diagram, Communication Diagram, Timing Diagram, and Interaction Overview Diagram.

The hierarchy of the diagrams is described in the next figure.

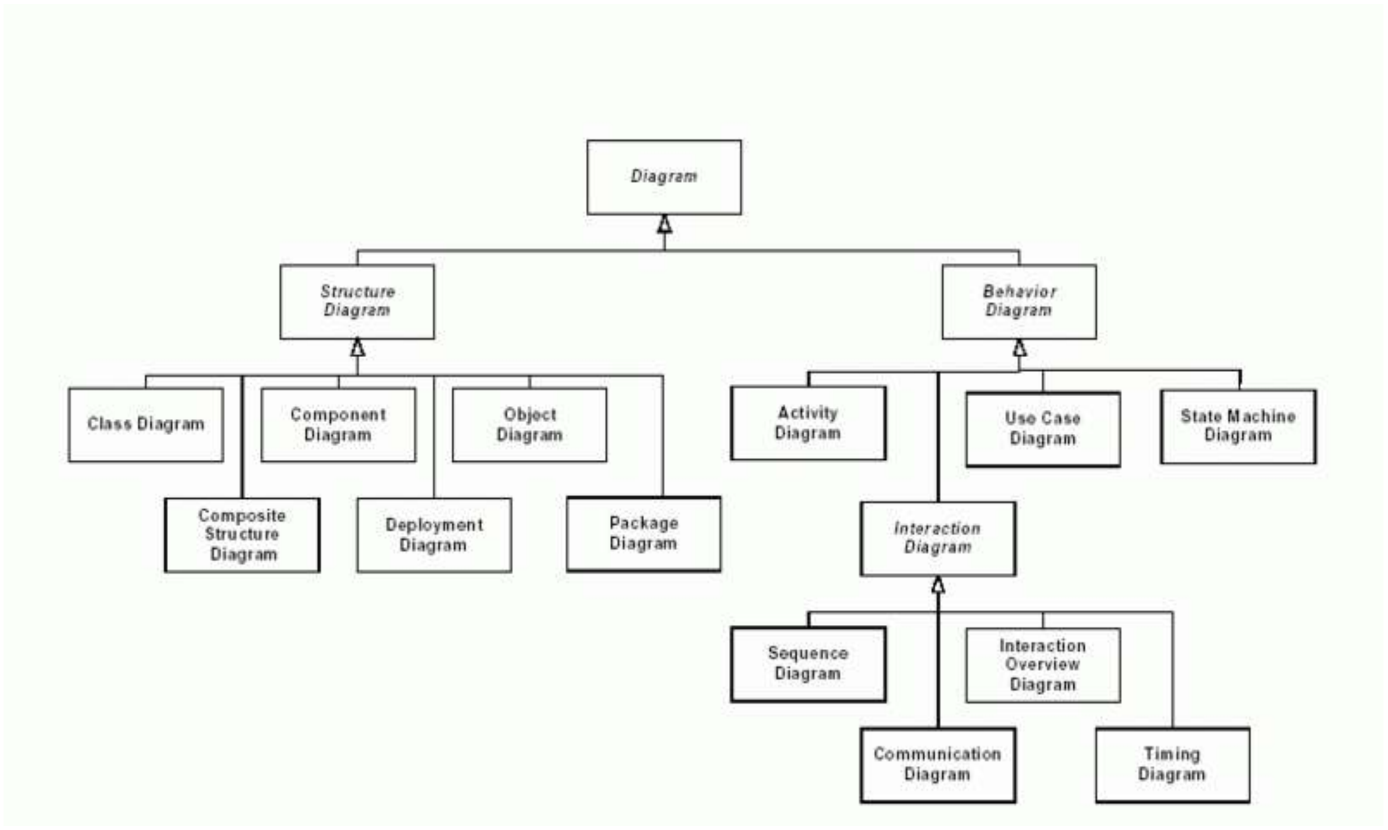


Figure 14: Hierarchy of UML Diagrams

UML can be considered as a modeling method for object-oriented systems and although the focus is on software and system development, UML is also considered as a possible method for modeling of business processes. Especially UML Activity Diagrams (ADs) provide a high-level means of modeling dynamic system behaviour.
The core element for the description of behaviour in UML is the Action. Actions take a set of input and convert them into a set of output, although either or both sets may be empty [11]. UML defines more than 40 action types (e.g. AcceptEvent or SendSignal). For the description of a system's behaviour the concept of activity is used, whereas an activity can be composed of various actions and/or other activities.
With Activity diagrams the basic control-flow concepts of the Workflow Management Coalition: sequence, parallelism, synchronisation, exclusive choice and simple merge can be modeled easily.

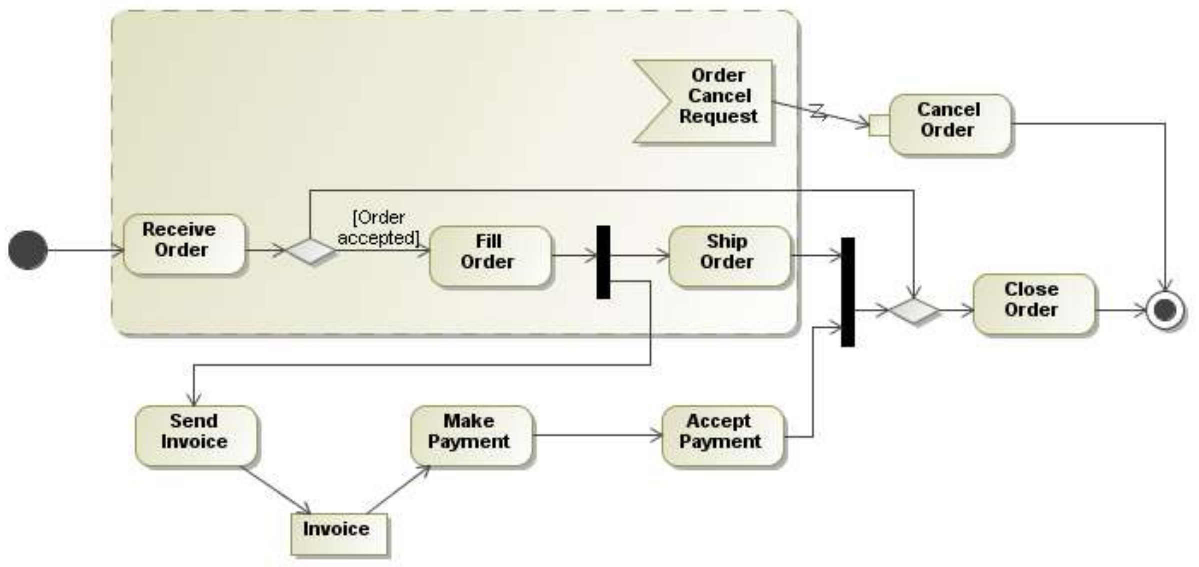


Figure 15: UML Activity Diagram Example

The above example shows some UML Activity Diagram actions (activities) and control nodes (initial/node, decision, fork, join).

UML Activity Diagrams also offer support for data perspectives like data visibility, data interaction, data transfer and data-based routing.

Given the large amount of existing tools with support for modelling with UML, both commercial and Open Source, does not seem necessary to present a specific list in this section.

3.2.2 Conclusions

UML specification is the most widespread software specification of OMG. The Activity Diagrams have been used for modeling business processes, but has a limited expressiveness. To solve this OMG reformed completely the activity diagrams to allow modeling all types of business processes. However OMG absorb BPMI (Business Process Management Initiative) organization and thus its standards, one of them was BPMN (Bussiness Process Modeling Notation), another notation language for modeling business processes, analyzed in the next section. This implies that there are two notations for the same purpose within a single organization, thus It appears that OMG are turning to the use of BPMN, as it is more expressive, giving support (total or partial) greatest number of workflow patterns and being richer graphically, making it easier to understand by different profiles of users. Another point to note is that BPMN is supported by the WfMC, one of the largest organizations in the field of workflows and a member of the OMG, itself has modified one of its specifications, XPDL, to give full coverage to BPMN.

Furthermore BPMN can be transformed directly into BPEL which is a language for Web service orchestration. BPEL is becoming recognized as a standard and will also be analyzed in this document.

Activity Diagrams pros are that their specification is supported by many tools and there are plenty of business processes modeled in this notation as well as a great experience by developers using UML diagrams.

3.2.3 BPMN 2.0

BPMN stands for Business Process Management Notation and represents a new standard developed by the Business Process Management Initiative (BPMI) and recently absorbed and maintained by the Object Management Group (OMG).

The OMG initiated the work to develop BPMN 2.0, the first major revision of BPMN in the year 2009. The beta version 2 of BPMN 2.0 is now available and vendors are using it to start implementations.

The main goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial process draft to the technical developers of the process executing applications; it is therefore thought to be used by different audiences. According to the BPMI the standard shall create a standardized bridge for the gap between business process design and process implementation [12]. Another main objective of BPMN is to make sure that XML languages designed for the execution of business processes (e.g., BPML4WS) can be visualized with a business-oriented notation to enable the direct mapping to business execution languages and web services. Thus, BPMN needs to allow for modeling with different methodologies, the creation of process segments as well as end-to-end business processes, and different levels of fidelity (from very simple to those ready for execution with a process engine). These requirements lead to two main drivers for designing the notation:

1) provide a simple and understandable mechanism for displaying business process models, and

2) provide the complexity inherent to business operations.

The approach taken to handle these two conflicting drivers was to organize the graphical aspects of the notation into a small set of notation categories (e.g., Activities, Events, and Gateways) so that the reader of a BPMN diagram can easily recognize the basic types of distinct elements and understand the diagram. Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look and feel of the elements.

The BPMN Modeling language offers different models type with a number of graphical objects and connecting objects. As one activity of a business process diagram can be decomposed to a sub-process, hierarchic process structures with different levels of detail can be modeled. If one activity has no further sub-process linked, it is considered a task.

3.2.3.1 BPMN 2.0 Structure

BPMN has more than 110 graphical symbols structured in several main categories, including Flow objects, Data, Connecting objects, swimlanes and Artifacts. From these elements, the specification then expands into layers that add specialized markers to these elements, this is because being a complex language it is structured in terms of extensibility layers, where each of them is based on the top and extends from the lower layers. It includes a core or kernel that includes the most fundamental elements of BPMN required to build BPMN diagrams: Process, Choreography, collaboration and conversation. The core is designed to be simple, concise and renewable, with a well-defined behavior.

Three layers are in charge to cover the different elements. Layer one contains the core elements grouped into three modules, infrastructure, common elements, used by layer two, and services (elements to cover the modeling of the services and infrastructures). Layer two defines extensions for three diagram types: process, choreography, and collaboration. And finally layer three defines extensions for humans, data, activities, and conversations.

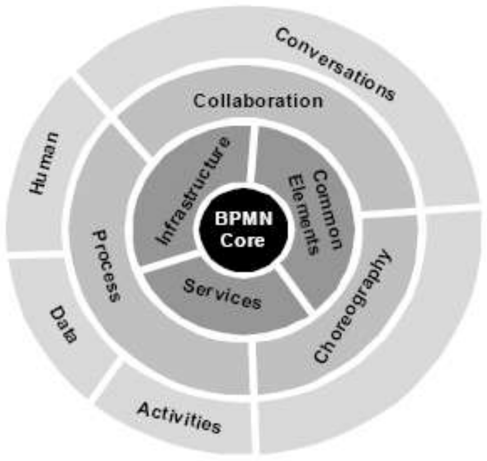


Figure 16: BPMN Layers Structure

The Core elements located in the first layer are:

- **Flow objects**

  Flow objects define the behavior of a business process and include events, activities and gateways

  - Activities: An activity can be generically described as work that an organization performs.

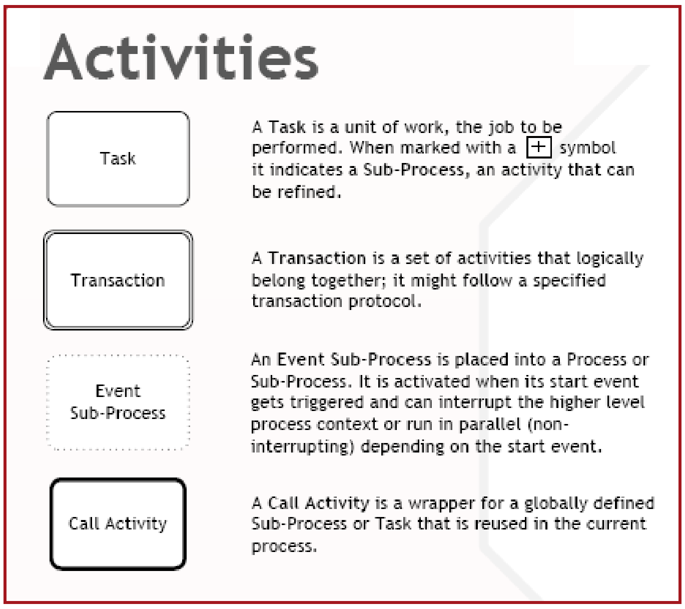

Activities
Task
A Task is a unit of work, the job to be performed. When marked with a ⊞ symbol it indicates a Sub-Process, an activity that can be refined.
Transaction
A Transaction is a set of activities that logically belong together; it might follow a specified transaction protocol.
Event Sub-Process
An Event Sub-Process is placed into a Process or Sub-Process. It is activated when its start event gets triggered and can interrupt the higher level process context or run in parallel (non-interrupting) depending on the start event.
Call Activity
A Call Activity is a wrapper for a globally defined Sub-Process or Task that is reused in the current process.

Figure 17: Event types in BPMN 2.0 [13]

Different elements for abstraction are remarkable:

Callable element : CallableElement is the abstract super class of all Activities that have been defined outside of a Process or Choreography but which can be called (or reused) from within a Process or Choreography. It may reference Interfaces that define the service operations that it provides. A Callable element could be exposed as a Service.

Call Activity : A Call Activity identifies a point in the Process where a global Process or a Global Task is used. The Call Activity acts as a 'wrapper' for the invocation of a global Process or Global Task within the execution. The activation of a call Activity results in the transfer of control to the called global Process or Global Task.

Global Task : A Global Task is a reusable, atomic Task definition that can be called from within any Process by a Call Activity.

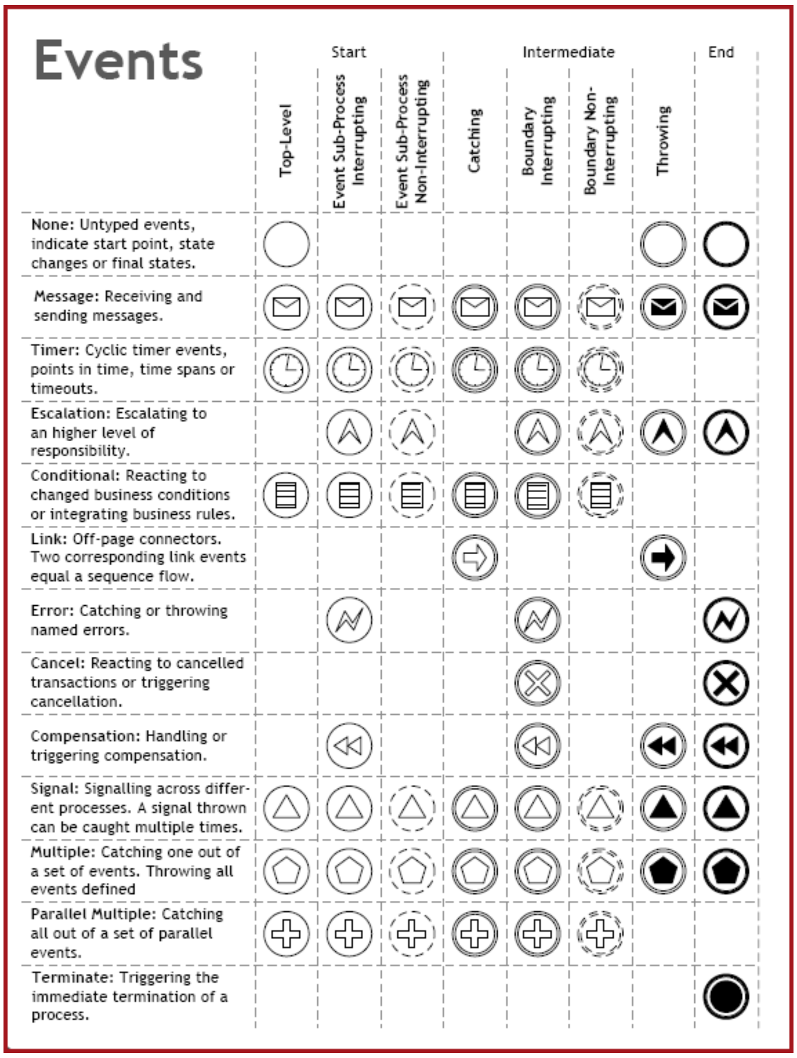
Also new elements have been added to improve dynamicity:

Business Rule Task : A Business Rule Task provides a mechanism for the Process to provide input to a Business Rules Engine and to get the output of calculations that the Business Rules Engine might provide. The InputOutputSpecification of the Task will allow the Process to send data to and receive data from the Business Rules Engine.

Service Task : A Service Task is a Task that uses some sort of service, which could be a Web service or an automated application. The Service Task inputs map to the parts of the input Message, that is the attributes inside of the Message. For a WSDL message, this would be expressed as message parts

- Events: Events represent something that "happens" during the course of a business process. An event will affect the flow of a process. An event is usually triggered by a cause and it will have an impact (result).

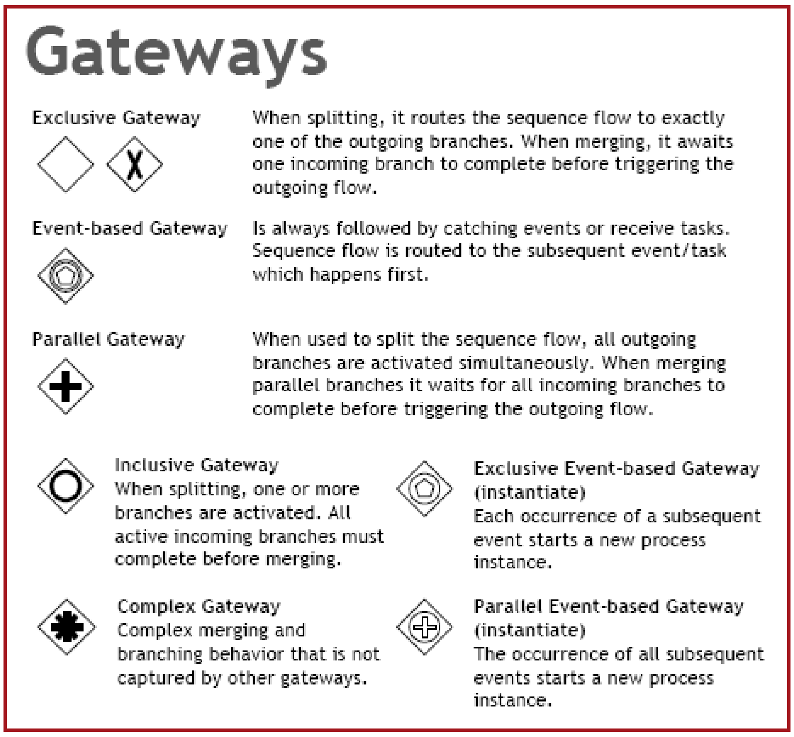
  There are three kinds of events: Start, Intermediate and End.



Events	Start			Intermediate				End
	Top-Level	Event Sub-Process Interrupting	Event Sub-Process Non-Interrupting	Catching	Boundary Interrupting	Boundary Non-Interrupting	Throwing	
None: Untyped events, indicate start point, state changes or final states.	○						○	○
Message: Receiving and sending messages.	✉	✉	✉	✉	✉	✉	✉	✉
Timer: Cyclic timer events, points in time, time spans or timeouts.	🕒	🕒	🕒	🕒	🕒	🕒		
Escalation: Escalating to an higher level of responsibility.		⋀	⋀		⋀	⋀	⋀	⋀
Conditional: Reacting to changed business conditions or integrating business rules.	▤	▤	▤	▤	▤	▤		
Link: Off-page connectors. Two corresponding link events equal a sequence flow.				⇨			➡	
Error: Catching or throwing named errors.		↯			↯			↯
Cancel: Reacting to cancelled transactions or triggering cancellation.					⊗			⊗
Compensation: Handling or triggering compensation.		⏪			⏪		⏪	⏪
Signal: Signalling across different processes. A signal thrown can be caught multiple times.	△	△	△	△	△	△	▲	▲
Multiple: Catching one out of a set of events. Throwing all events defined	⬠	⬠	⬠	⬠	⬠	⬠	⬟	⬟
Parallel Multiple: Catching all out of a set of parallel events.	✚	✚	✚	✚	✚	✚		
Terminate: Triggering the immediate termination of a process.								●

Figure 18: Event types in BPMN 2.0 [13]

- Gateways: are like decisions and can be exclusive, inclusive, complex or parallel



Gateways

Exclusive Gateway
When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.

Event-based Gateway
Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.

Parallel Gateway
When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.

Inclusive Gateway
When splitting, one or more branches are activated. All active incoming branches must complete before merging.

Exclusive Event-based Gateway (instantiate)
Each occurrence of a subsequent event starts a new process instance.

Complex Gateway
Complex merging and branching behavior that is not captured by other gateways.

Parallel Event-based Gateway (instantiate)
The occurrence of all subsequent events starts a new process instance.

Figure 19: Gateway types in BPMN 2.0 [13]

- **Data**

  Data include five elements and corresponds to the data input and output for the different process or tasks and also for the data repository.

  - Data Objects
  - Data Imputs
  - Data Outputs
  - Data Stores
  - Properties

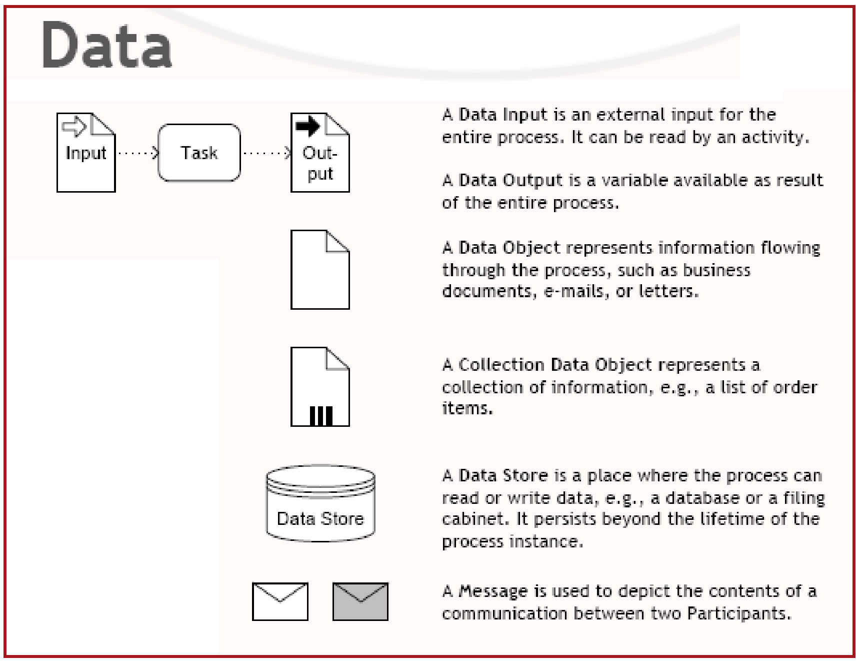

Data
Input
Task
Out-put
A Data Input is an external input for the entire process. It can be read by an activity.
A Data Output is a variable available as result of the entire process.
A Data Object represents information flowing through the process, such as business documents, e-mails, or letters.
A Collection Data Object represents a collection of information, e.g., a list of order items.
Data Store
A Data Store is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.
A Message is used to depict the contents of a communication between two Participants.

Figure 20: Data elements in BPMN 2.0 [13]

- **Connecting Objects**

  The connecting objects can be sequence flows but it is also possible to model additional message flows, especially in order to represent complex B2B-scenarios. Data objects which can be both electronic and physical objects can be linked to sequence flows or message flows. Different ways to connect the flow objects are through: *Sequence flow, Message flow, Association* and *Data Association*

- **Pool and Swimlanes**

  These two graphical elements, *pools* and *lanes* specify within a process which element does what. Normally pools are used for Modeling organizations and lanes which are sub-partitions of pools for departments within that organization although a pool could also represent a function, application or location, among other things.

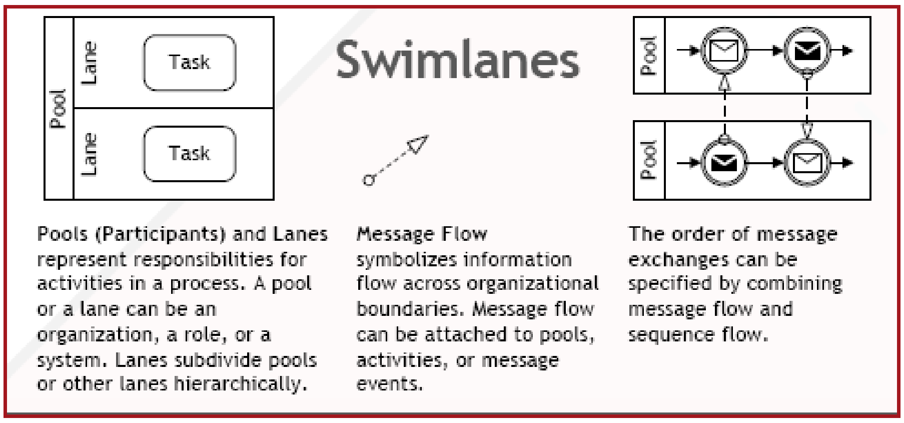

Swimlanes
Pool
Lane
Task
Lane
Task
Pool
Pool
Pools (Participants) and Lanes represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically.
Message Flow symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events.
The order of message exchanges can be specified by combining message flow and sequence flow.

Figure 21: Pool and Swimlanes in BPMN 2.0 [13]

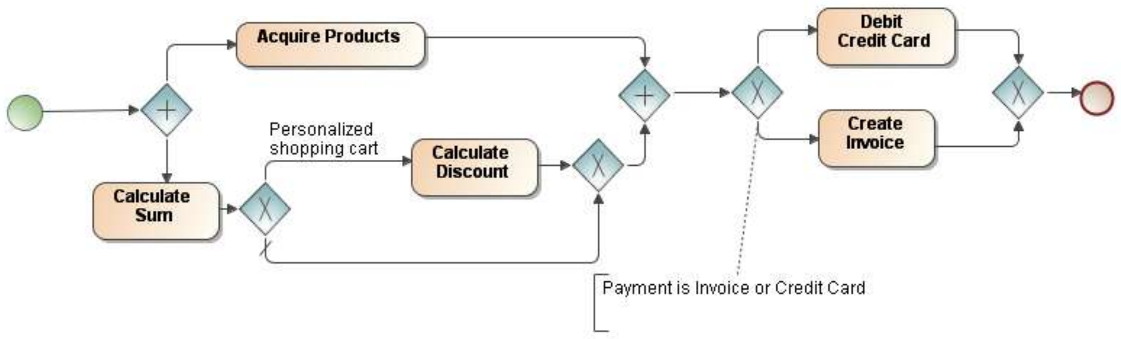
- **Artifacts**

  Provide additional information about the process and is composed by *Text annotations* and *Groups*.

3.2.3.2 BPMN 2.0 Diagrams

The BPMN models are expressed graphically through different types of diagrams, in BPMN 2.0, four types of diagrams are available:

Process: Process provides a flowchart view that describes a sequence or flow of Activities in an organization. In BPMN a Process is a graph of flow elements, which are a set of activities, events, gateways, and sequence flow. BPMN supports the modelling of both, simple process and processes with more complex concepts as transactions, compensations, etc..



,

Figure 22: Business Process diagram (Example)

Collaboration: Collaboration diagram provides a view of the interactions shown by *message Flow* between two or more participants shown as Pools. Collaborations can be combined with Processes within the Pools to show how the interactions are related to the internal Process activities and/or with Choreographies between the Pools.

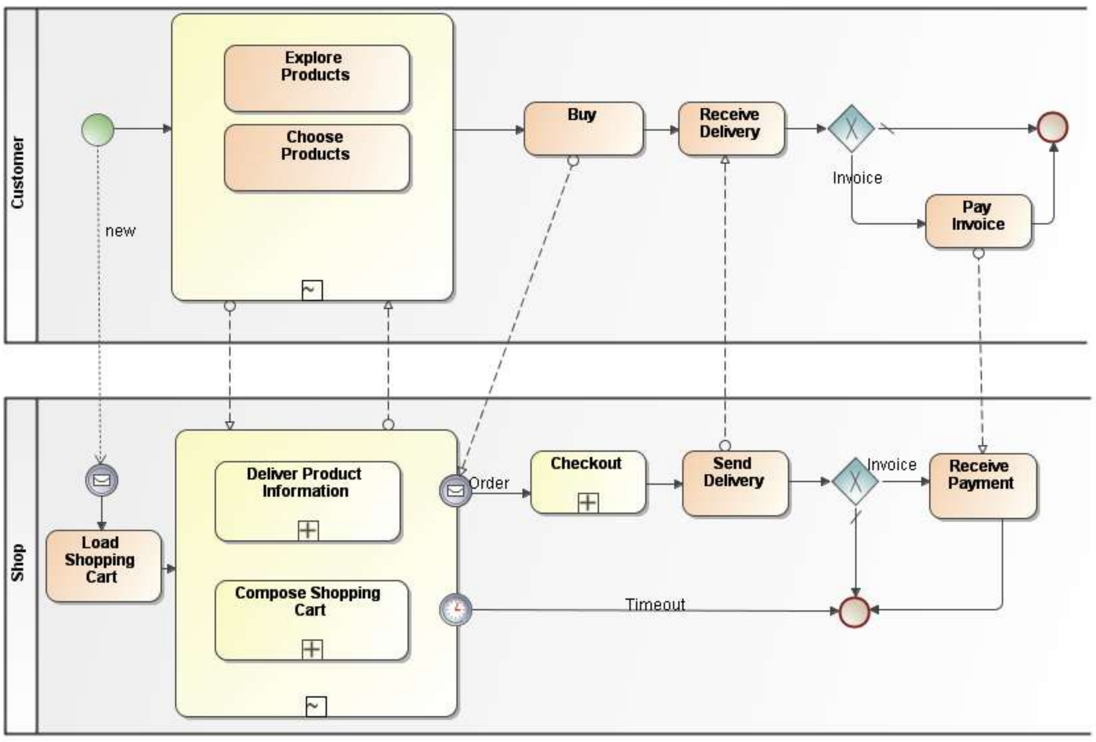


Figure 23: Business Collaboration Diagram (Example)

Conversation: Conversation defines message exchanges between participants/Pools and allows a modeller to group Collaboration interactions between two or more Participants/Pools, which together achieve a common goal.

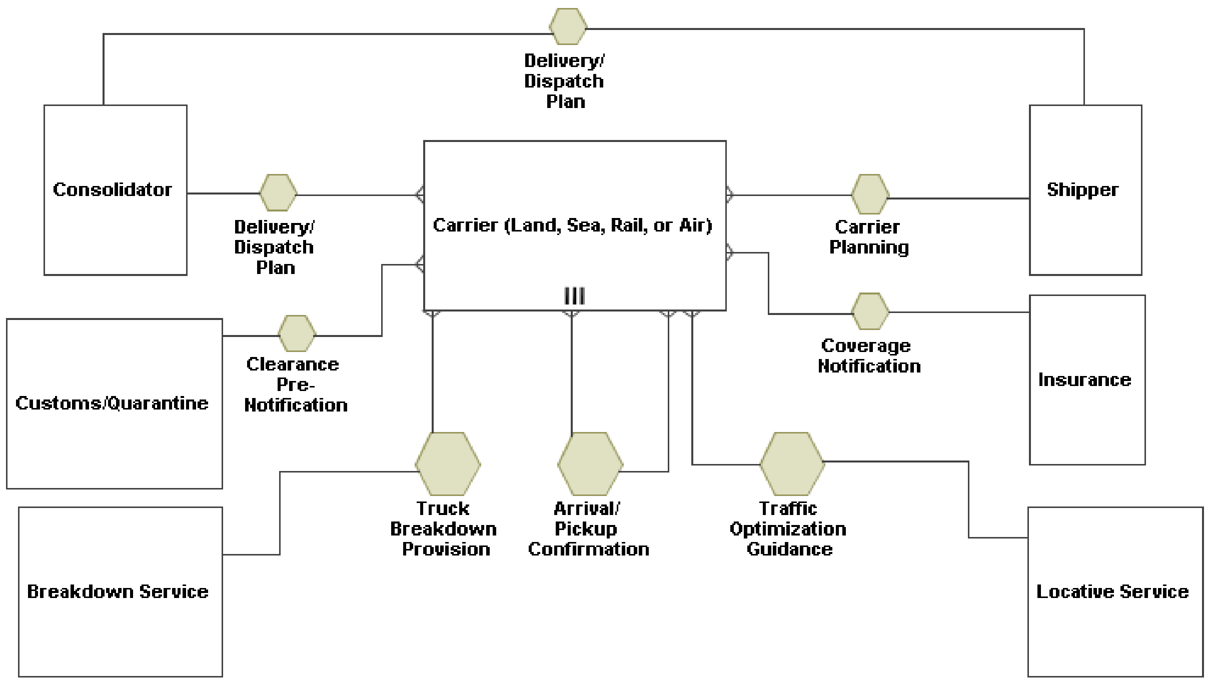


Figure 24: Conversation Diagram (Example)

Choreography: Choreographies represent sets of tasks performed by participants. This kind of diagram provides a flow chart view to sequence the interactions between the participants. The focus is on the exchange of information and definition of the messages.

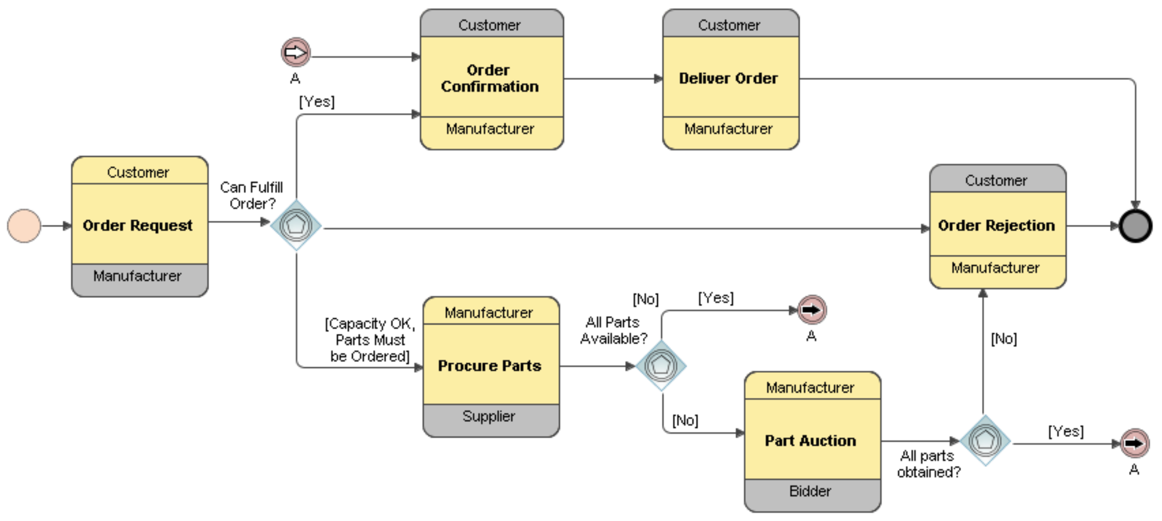


Figure 25: Choreography Diagram (Example)

BPMN 2.0 represents a substantial effort to improve the underlying modeling infrastructure and to expand the modeling capabilities to cover a wider range of business process scenarios and interactions.

3.2.3.3 Conclusions

BPMN 2.0 represents a substantial effort to improve the underlying modeling infrastructure and to expand the modeling capabilities to cover a wider range of business process scenarios and interactions. This last specification offer real improvements about the refinement and formalization of the BPMN execution semantics, they stand at the beginning of a process and at the same time offer an important Interchange of BPMN models, including the diagram layout, through XML and XMI Schemas (and XSLT transformation) , thus a set of conformance levels to support different process modeling markets.

New process elements in the standard make more understandable and useful. It includes:

- Non-Interrupting Events (attached to Activities)
- Optional Event Sub-Processes
- Graphical markers for individual Task types
- Improved support for human interactions

- Enhanced modeling of services
- Upgrade support for process data
- The Refinement of Event composition and correlation and
- New diagrams, Conversation and Choreography

For all the abovementioned bullets this standard is consolidating as the referent in the market.

3.2.3.4 Tools

Different tools/editors Open Source and Commercials offering the BMPN 2.0 version, some of the best-known in the market are:

- **Oryx online editor**

  It is an Open Source project that allows to add prototypes to a powerful process modeling infrastructure. The project is mainly driven by the Business Process Technology research group.

  Source: http://bpt.hpi.uni-potsdam.de/Oryx

- **Eclipse BPMN modeler**

  The BPMN Modeler is a business process diagram editor for business analysts, based in eclipse and Its semantic model is based on the Eclipse Modeling Framework, it uses the Graphical Modeling Framework to run its editor.

  Source: http://www.eclipse.org/bpmn

- **Activiti BPMN 2.0 Eclipse Plugin**

  The Activiti Eclipse BPMN 2.0 Designer eclipse plugin project provides the necessary functionality to design BPMN 2.0 processes and run these processes on the Activiti Engine.

  Source:
http://docs.codehaus.org/display/ACT/Activiti+BPMN+2.0+Eclipse+Plugin

A complete list of tools that implements BPMN collected by OMG is in:

http://www.omg.org/bpmn/BPMN_Supporters.htm

3.2.4 Event Driven Process Chain (EPC) – ARIS

Event-driven Process Chains are a method developed by Scheer, Keller and Nüttgens within the framework of Architecture of Integrated Information Systems (ARIS) [14] to model business processes especially for the support of business users.

The ARIS concept involves dividing complex business processes into separate views and integrating these separate views to form a complete overview about one business process. The different views are:

- **Function view**: Describes the activities within a company which are to be performed, the enumeration of the individual subfunctions that belong to the overall relationships and the relationships between the functions.
- **Data view:** Contains events and status information. Events are created by processing functions or by actors outside of the model. An event may act as a pre- or postcondition of a function.
- **Organization view**: For the structure and relationships between users and organization units which are responsible for performing a function.
- **Resource view**: which includes general conditions for describing other components and deliverables that represent services or products that functions produce or need.

The EPC method includes three core elements: events, functions and connectors. Events represent states created by processing functions or by actors not included in the model. Functions are like activities. So a process consists of a sequence of events triggering business functions which are themselves the result of other functions or an initial triggering event. The semantic for the connectors is designed to be able to model complex parallelisms and decisions:

- **AND-connector**: one incoming arc and at least two outgoing arcs mean that that process is forking into various parallel activities, whereas various incoming arcs and only one outgoing arc represent the merging of the parallel process paths.

- **OR-connector**: one incoming arc and at least two outgoing arcs mean that the process is forking into alternative functions which are not necessarily exclusive, whereas various incoming arcs and only one outgoing arc represent that two or more possible paths can lead to the next function.
- **XOR-connector**: like the OR-connector but with exclusive paths (only one alternative can be taken).

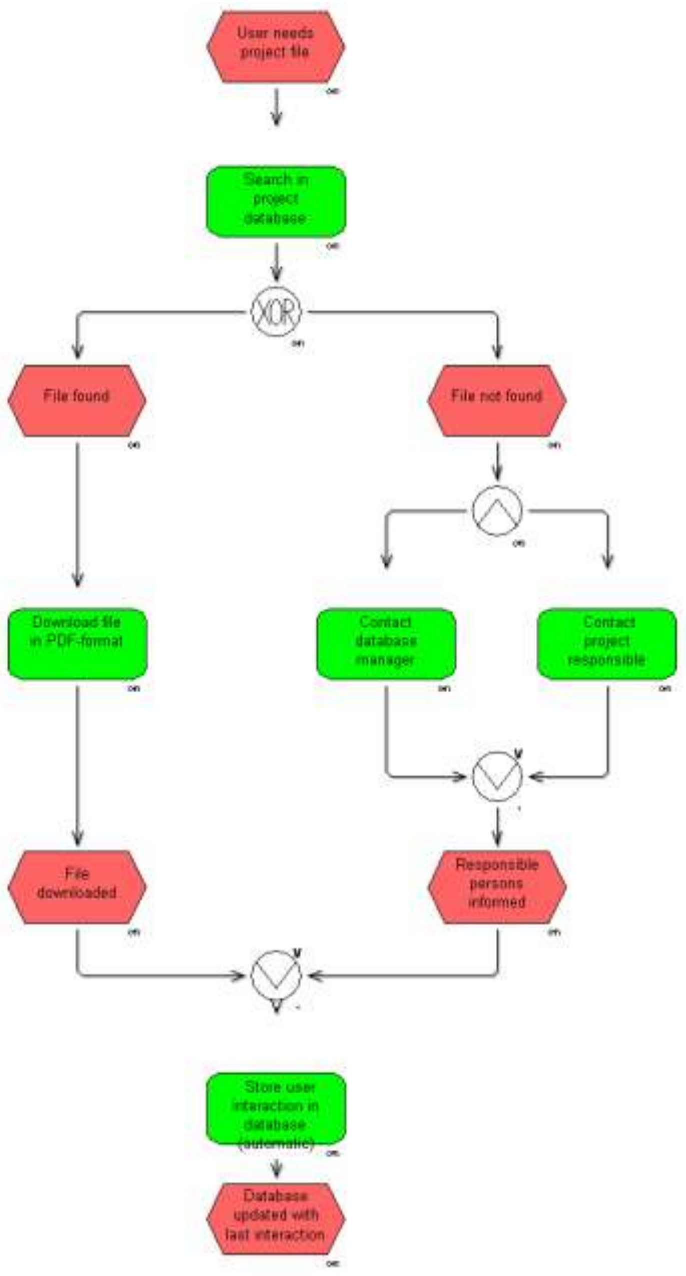

User needs project file
Search in project database
XOR
File found
File not found
Download file in PDF-format
Contact database manager
Contact project responsible
File downloaded
Responsible persons informed
Store user interaction in database (automatic)
Database updated with last interaction

Figure 26: EPC example process

EPC processes can be modelled in a hierarchical structure linking functions to sub-processes. The basic process flow modelled with EPC can be extended by further semantic components like data flows, organizational units or IT systems.

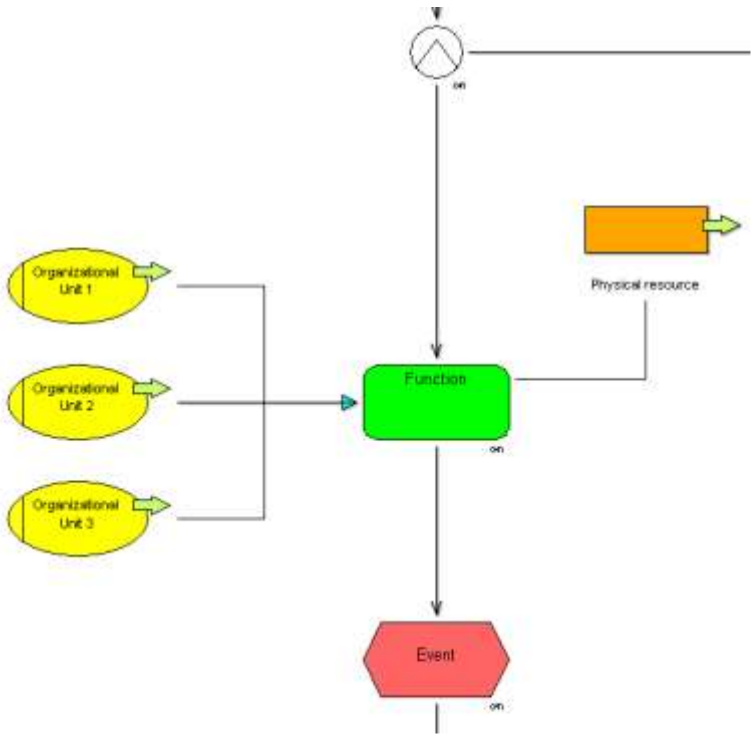


Figure 27: EPC with additional elements

Although the EPC-method has a clear focus on process analysis and design rather than on technological aspects.

3.2.4.1 Conclusions

ARIS is a complex and widespread methodology whose notation is more used in a vast range of issues relating to development, optimization, integration and implementation of an information system. Currently is having considerable success in the field of business processes, despite being a commercial tool and its high price. The most important features is that the EPC diagrams are understandable to non-specialists in the modeling process, and at the same time allows a multilevel view of the processes, with the possibility of high-end models and levels with highly detailed, offering technological support for collaborative development.

About the expressiveness, EPC diagrams can be ambiguous and have no well-defined syntax. The absence of a well-defined formal semantics is a major impediment to the exchange of models between tools from different vendors, developing tools for process analysis and prevents a study precise on the workflow patterns to which it gives those who supported this diagram. One solution might be to try to formalize the EPC diagrams transforming them in other formalized notation as Petri nets, not covered in this analysis mainly because is difficult to understand for non-experts and presents problems in expressing some patterns related to specific features such as multiple instances, advanced synchronization, cancellation, etc.

3.2.5 IDEF

IDEF (Integrated Definition Methods) is a family of standard methods for the documentation and analysis of business processes sponsored by the US Air Force within the context of its long-running Integrated-Computer-Aided-Manufacturing-Program. IDEF is a set of Modelling methods with IDEF0 and IDEF3 being the most popular ones. The original aim of IDEF was to cover the need of comprehensive communication techniques to describe complex business processes (especially with respect to manufacturing).

3.2.5.1 IDEF0

IDEF0 is a method for functional modelling with the objective of describing a system’s functions in a detailed way. An IDEF0 diagram is a graph where the nodes (represented by boxes) represent functions and the directed edges represent data and control flows between the functions.

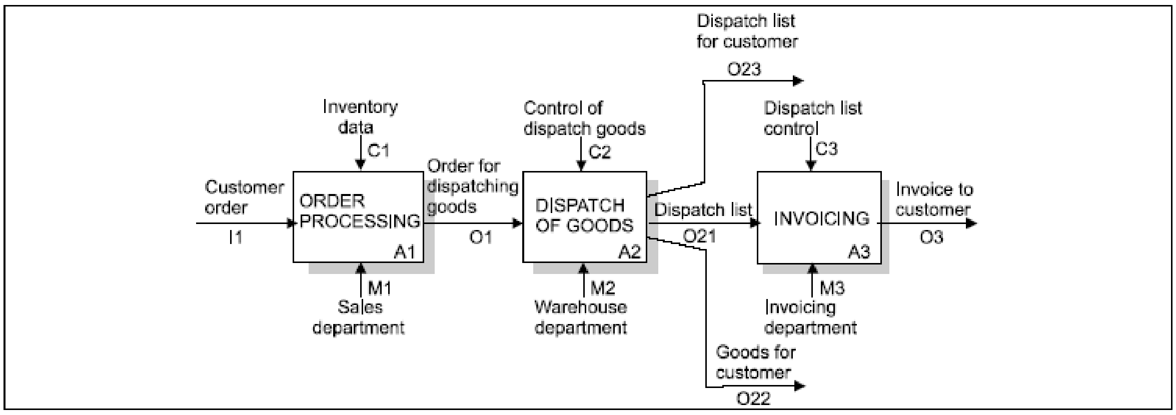


Figure 28: IDEF0 (Example)

IDEF0 only represents the functional view of a business process and was originally based on techniques for software development and design.

3.2.5.2 IDEF3

IDEF3 shows the dynamic aspects of a business process or the behavioural view. Processes are described as ordered sequences and as such, IDEF3 is a scenario-driven process flow modelling method, based on the direct capture of precedence and causality relations between situations and events.

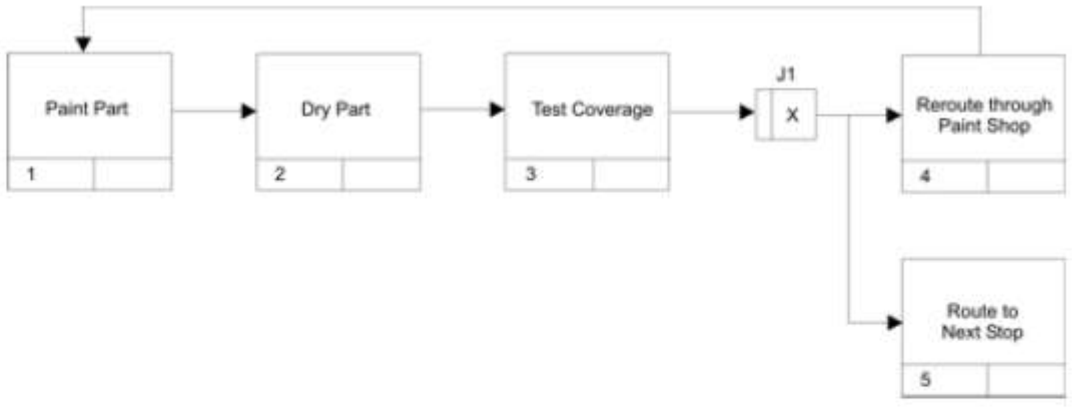


Figure 29: IDEF3 – Process Description Diagram (Example)

The IDEF3 term for elements represented by boxes (can be activities, processes of events) is Unit of Behaviour (UOB). Each UOB can have associated with it descriptions in terms of other UOBs (decomposition) or descriptions in terms of a set of participating objects and their relations (elaboration). Decomposition is something like a sub-process or sub-model whereas an elaboration is an element of the IDEF3 description which captures the objects that participate in a particular activity and the corresponding constraints.

Another model type in IDEF3 is the Object State Transition Network (OSTN) diagram which shows object-centred views of processes and summarizes the allowable transitions.

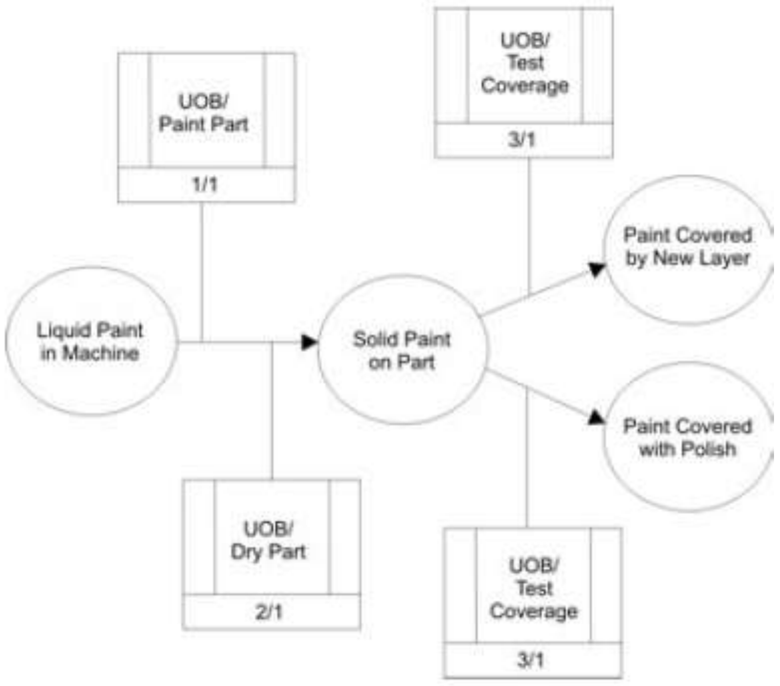


Figure 30: IDEF3 – Object State Transition Network Diagram (Example)

Object states are represented by circles and state transitions are represented by the lines connecting the circles. An object state is defined in terms of the facts and constraints that need to be true for the continued existence of the object in that state and is characterized by entry and exit conditions. Entry conditions give the requirements to be met before an object can transition into the next state, while exit conditions give the requirements for transitioning out of a state [15].

3.2.5.3 Conclusions

IDEF0 is a powerful methodological tool and has been used in the aerospace, electronics industry, pharmacy , and fast moving consumer goods fields. IDEF0 is a simple notation (based on boxes and lines) which allows to model activities incorporating flow data in and out of them, as well as the business rules and roles, incorporated all in the same view. However IDEF0 model does not reflect correctly the interactions among members of the team, but offers the possibility of combining with other methodologies to add sequencing and timing of activities.

IDEF0 is an appropriate model to be used whenever it you need to generate processes with a high degree of accuracy and detailed models in the description.

IDEF3 allows to document processes for standardization and for capture the temporal sequence and decision that affects the process logic. It is also used to analyze existing processes and to design and test new processes before their implementation.

Ideally, would be to use jointly IDEF0 & IDEF3 representing details of deployment as well as the processes at the appropriate level for each moment.

About expressiveness, IDEF0 e IDEF3 supports almost all workflow patterns, but there are deficiencies to reflect organizational structures and aspects related to the objectives and the qualitative characteristics of the process.

3.2.6 BPML

Business Process Modelling Language (BPML) is an open specification defined by BPMI. It aims to enable the standards-based management of e-Business processes with forthcoming Business Process Management Systems (BPMS).

The Business Process Modelling Language (BPML) is a meta-language for the modelling of business processes, just as XML is a meta-language for the modelling of business data. BPML provides an abstracted execution model for collaborative transactional business processes based on the concept of a transactional finite-state machine. BPML considers e-Business processes as made of a common public interface and as many private implementations as process participants. This enables the public interface of BPML processes to be described as ebXML business processes, independently of their private implementations. In much the same way XML documents are usually described in a specific XML Schema layered on top of the eXtensible Markup Language, BPML processes can be described in a specific business process modelling language layered on top of the extensible BPML XML Schema. BPML represents business processes as the interleaving of control flow, data flow, and event flow, while adding orthogonal design capabilities for business rules, security roles, and transaction contexts. Defined as a medium for the convergence of existing applications toward process-oriented enterprise computing, BPML offers explicit support for synchronous and asynchronous distributed transactions, and therefore can be used as an execution model for embedding existing applications within e- Business processes as process components.

BPML describes the structural representation of a process and the semantics of its execution. As with BPEL, the vision of BPML is to run XML processes on an engine element by element, according to precisely defined semantics. The code of a BPML process with familiar constructs such as loops, decisions, parallel paths, variables, and structured exception handling is readily understood by a programmer.

The essential language constructs to create a BPML process are :

- **The basic process structure**: BPML processes are enveloped in a package. Each process has a name, a set of activities, and optionally, a compensation handler. A process can have subprocesses, as well as a context, which in turn can contain fault handlers, exception processes, subprocesses, and properties. There are three ways to start a process: through an activity, a message, or a signal.
- **Variables and assignments**: These are called properties, which can be declared at the scope of a package or a context.
- **Exception handling and compensation**: BPML contemplate three error handling approaches:
  - An exception process: Iis an event handler that aborts a process or a set of activities.
  - A fault handler: Catches and attempts to fix an error that occurs in an activity.
  - A compensation process: Is like an exception process, except it is intended to revert a completed, rather than an in-flight, process; additionally, a compensation process is defined for a process, whereas an exception process is defined for a given context. Compensation is triggered by a call to the activity compensate.
- **Split and join**: If one activity executes a set of actions in parallel and merges its results, sometimes are called the AND split and join. If some actions are executed in parallel; when the last of them completes the process continues with its next step. The switch activity, sometimes called the XOR split and join, executes exactly one action from a set. Inside the switch activity is a sequence of case statements, each of which has a condition and associated activity. The activity to be executed is the one defined in the first case statement whose condition evaluates to true. If no case's condition is true, the activity belonging to the default case is run..
- **Loops**: In BPML three types of loops are supported, each defined as a complex activity type: while, until, and foreach.

  - **while** executes a set of activities zero or more times, checking a Boolean-valued condition at the beginning of the loop to determine whether to continue.
  - **until** is similar, but it checks the condition at the end of the loop.
  - **foreach** loops through a list of items, executing a set of activities for each item in the list.

- **Participant exchange**: A BPML process can exchange messages with external participants through web services in its action and choice activities. action can either call a web service or implement a web service that, when called, triggers an event in the process. Whereas BPEL cleanly separates these operations into invoke, receive, and reply. BPML overloads the action activity to serve all three purposes. An action is specified with a name, a WSDL port type and operation, a set of inputs and/or outputs (mapping to WSDL input and output messages types, respectively, for the given operation), and an optional correlation expression, which filters action events to trigger only for given input values.
- **Transactions**: BPML is interoperable with open transaction standards such as WS-Transaction. BPML defines a special fault type, bpml:rollback, that a process can use to trigger rollback handlers
- **Extensions**: BPML supports adding custom attributes or elements to most of its elements.
-

3.2.6.1 Conclusions

BPML emerged as one of the first specifications for modeling; the basic idea was of creating a conceptual standard for defining processes.

BPML doesn't specify any business process semantics like Activities or Work Items and neither the metamodel includes the concept of Role, meaning that it is not possible to specify the role that performs a given activity. A logical business partner, system component or user are examples of roles in the general process definition. The role could be simulated using locator attribute defined in BPML specification's activity types.

Currently this standard as well as the entity that developed, BPMI has disappears absorbed by OMG in 2005. BPML disappeared, supplanted by BPEL, another standard analyzed in a later section. For these precise reason it will not be reflected in the comparative table, but has been analyzed because was the precursor of all the modeling business processes languages.

4. Workflow and Business Process Execution Languages

Workflow and Business Process execution languages in a rough way are focused in the process control across control flow instructions and with the execution of the processes, in this section three of them are analyzed: XPDL, YAWL and BPEL.

4.1 XPDL 2.1

XML Process Definition Language (XPDL) [16] is the language proposed by the Workflow Management Coalition (WfMC) to interchange business process definitions between different workflow products. The goal of XPDL is to provide a common language for the workflow domain allowing for the import and export process definitions between a variety of tools ranging from workflow management systems to modeling and simulation tools.

In April 2008, the WfMC ratified XPDL 2.1 as the last revision of this specification. Before that, the first specification, XPDL 1.0 was ratified by the WfMC in 2002, and in October 2005 was ratified XPDL 2.0. Currently XPDL 2.2 specification supporting BPMN 2.2 is almost finished and ready to delivery. XPDL 2.1 includes extension to handle BPMN 1.1 constructs, as well as clarification of conformance criteria for implementations. With the publication of the latest BPMN 2.0 specification a new version of XPDL specification (version 2.2) is almost finished to be published.

The WfMC has identified five functional interfaces to a workflow service as part of its standardization program. XPDL specification forms part of the documentation relating to "Interface one", supporting Process Definition Import and Export of process definitions. In this way XPDL defines an XML schema for specifying the declarative part of workflow / business process and to store and interchange business process definitions addressing both the graphics and the semantics [17].

XPDL is strongly linked with BPMN Standard and provides a file format that supports every aspect of the BPMN process definition notation including graphical descriptions of the diagram, as well as executable properties used at run time. This allows interchanging diagrams between different tools and reproducing the diagram with totally fidelity.

The main elements of the language are:

- **Package:** Element which is the container holding the other elements.
- **Application:** The Application element is used to specify the applications/tools invoked by the workflow processes defined in a package.

- **Workflow-Process:** The element WorkflowProcess is used to define workflow processes or parts of workflow processes. A WorkflowProcess is composed of elements of type Activity and Transition.

- **Activity:** The Activity element is the basic building block of a workflow process definition. Elements of type Activity are connected through elements of type Transition. Basic activities have attributes which provide information about who can perform the activity, what application or web services should be invoked, what properties of the object being worked on are used and/or altered in this step, and so forth.

  There are three types of activities:

  - Route: This type is dummy activities just used for routing purposes.
  - Implementation: Activities of type Implementation are steps in the process which are implemented by manual procedures, implemented by one of more applications (Tool), or implemented by another workflow process (Subflow).
  - BlockActivity: Type used to execute sets of smaller activities.

  There are seven standard Tasks that can be specified for a basic Activity and are used primarily for invoking Web Services and using WSDL messaging. An eighth task type is used for invoking Applications whose signatures have been defined in the Business Process Model.

  Applications that can be invoked by a process are defined at the Process or Package (See figure 32 - Package metamodel) level. There are multiple types of applications:

  - Traditional applications
  - Components
  - Web Services
  - Business Rules
  - Form
  - Script

- **Transition:** Element used for connecting activities and indicating a transition between these.
- **Participant:** The Participant element is used to specify the participants in the workflow, i.e., the entities that can execute work. There are 6 types of participants: ResourceSet, Resource, Role, OrganizationalUnit, Human and System.
- **DataField, DataType:** Elements used to specify workflow relevant data. Data is used to make decisions or to refer to data outside of the workflow, and is passed between activities and subflows.

4.1.1 XPDL Metamodel

XPDL describes the meta-model, which is used to define the objects and attributes contained within a process definition. The XPDL grammar is directly related to these objects and attributes. For this approach two operations are provided by a vendor:

• Import a workflow definition from XPDL.

• Export a workflow definition from the vendor's internal representation to XPDL.

XSL Stylesheets can be used by the vendor for both operations.

The Meta-Model describes the entities at the highest level in the domain of the definition of processes, their relationships and attributes of processes (including some which may be defined for simulation or monitoring purposes rather than for enactment). It also defines various conventions for grouping process definitions into related process models and the use of common definition data across a number of different process definitions or models.

For all these aspects XPDL has two main meta-models, described in the following subsections.

4.1.1.1 Package Metamodel

Package Metamodel identifies the entities and attributes for the exchange, or storage, of process models. The Package Definition allows the specification of a number of common process definition attributes, which will then apply to all individual process definitions contained within the package. Such attributes may then be omitted from the individual process definitions.

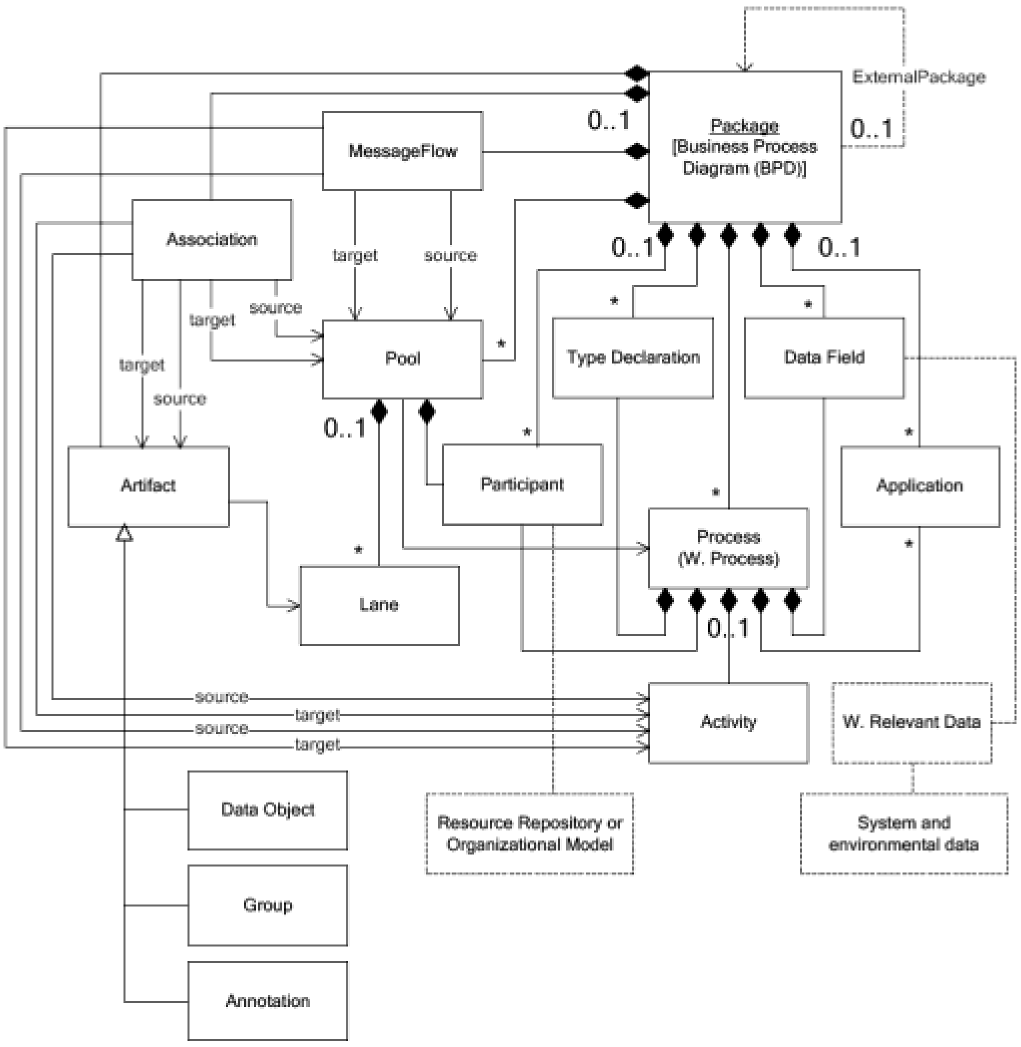


Figure 31: XPDL 2.1 Package Metamodel

4.1.1.2 Process Metamodel

The Process metamodel identifies the basic set of entities and attributes for the exchange of process definitions and depicts the relationships between all the elements in a Process. These entities contain attributes that support a common description mechanism for processes and must be defined for process definition. There are two ways to do so, explicitly at the level of the process definition, or by inheritance directly or via cross reference from a surrounding package.

The XPDL Process and WorkflowProcess have directly correspondence with the BPMN Process.

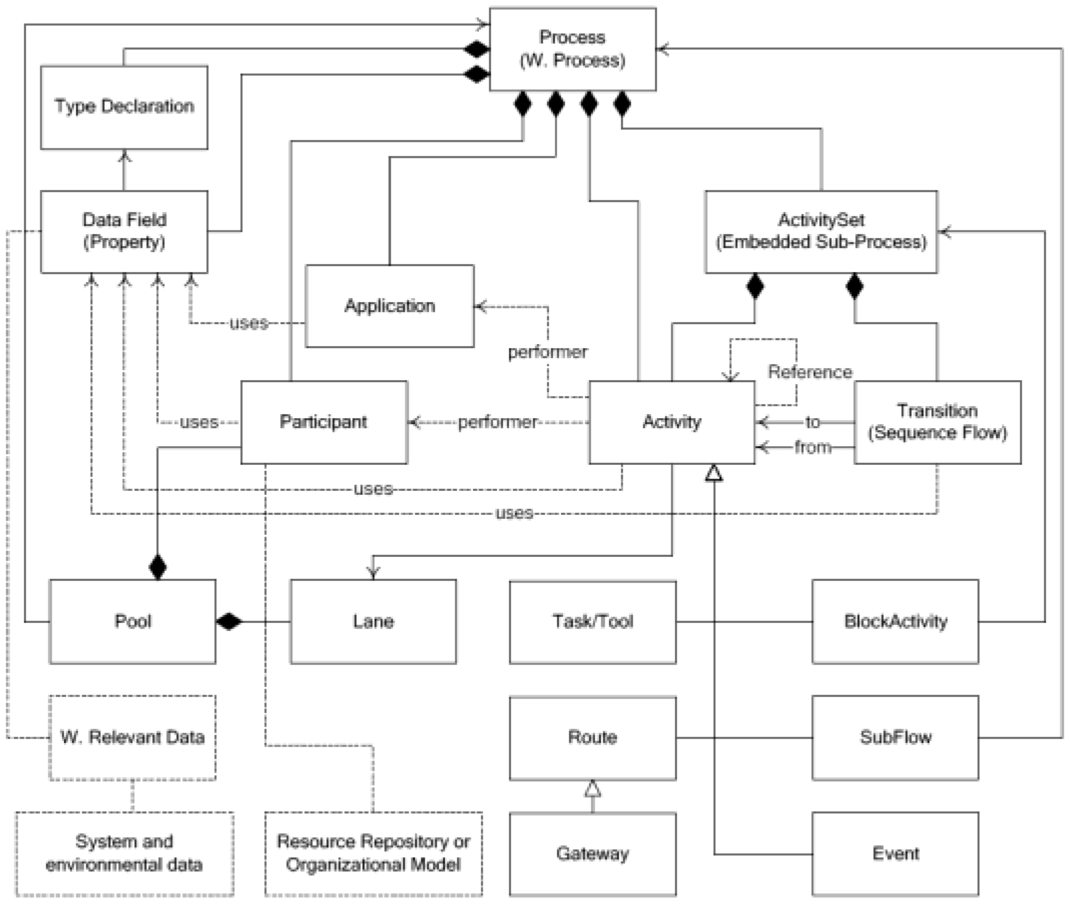


Figure 32: XPDL 2.1 Process Metamodel

4.1.2 Tools

Multiple tools Open Source and Commercials understand XPDL. In general are tools for modeling business processes and bring this feature. Some relevant in the market:

- **TIBCO® Business Studio**

  TIBCO Business Studio™ Community Edition is a free, standards-based, business process modeling environment that enables business experts to model and simulate business processes and their supporting data and organization models.

  Source: http://developer.tibco.com/business_studio/

- **Together XPDL Workflow Editor**

  Together XPDL Workflow Editor is a graphical Java Workflow Editor fully implementing WfMC.

  Source: http://www.together.at/prod/workflow/twe

- **BOC ADONIS 3.7**

  ADONIS is Commercial tool focused in business process modeling with support of different modelling standards and notations such as BPMN, UML, EPC, and LOVEM.

  Source: http://www.boc-group.com/products/adonis/

A complete list of all the tools recognized by WfMC is in:

http://www.wfmc.org/xpdl-implementations.html

4.1.3 Conclusion

XPDL 2.1 provides a standard file format for persisting BPMN diagrams and interchanging Process definitions; It supports every aspect of the BPMN process definition notation including graphical descriptions of the diagram, as well as executable properties used at run time. The file format is based on the WfMC meta-model which establishes a framework for defining, importing and exporting process definitions for numerous products including execution engines, simulators, BPA modeling tools, Business Activity Monitoring and reporting tools. The schema defining the format is extensible and provides vendor and user extension capabilities as well as a natural path for future versions of the standard. However this approach does not result in a common language and also even the semantics of the core constructs of XPDL remain undefined.

Basically XPDL is the Serialization Format for BPMN and currently can be considered the best file format for exchange of BPMN diagrams.

4.2 YAWL

Yet Another Workflow Language (YAWL) [17] despite not being supported by any standardization committee is interesting to be taken into account. YAWL is a workflow language based on the Workflow patterns developed and still evolving thanks to an open source community leaded by Eindhoven University of Technology and Queensland University of Technology and offer under the LGPL license.

The language is supported by a software system that includes an execution engine, a graphical editor, a worklist handler and an analisys and verification tool. On the basis of graphic files can produce .xml files with only Workflow process data for use in the execution engine.

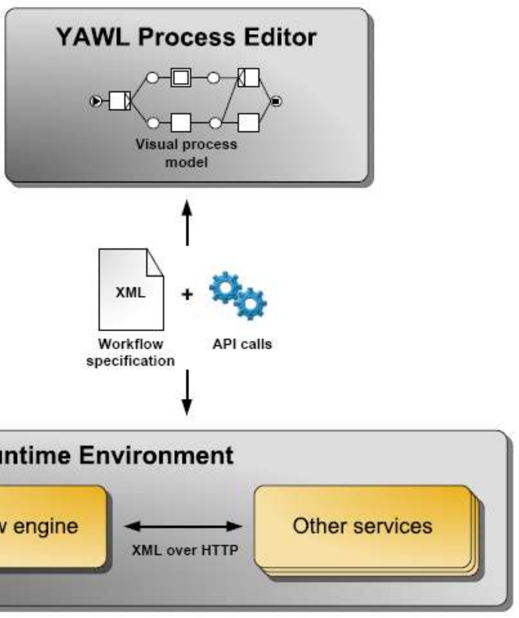


Figure 33: YAWL components

As before mentioned, YAWL is based on the Workflow patterns which are divided into different categories as control-flow patterns, resource patterns, data patterns and exception handling patterns.

Control flow pattern, YAWL metamodel definition involves tasks, that can be atomic, composite or multiple instance, then these patterns are used to understand how process flows from task to task within the workflow, and the kind of rules that can be embedded into the junctions between a task and the link that flows out of it, basically two main conditions elements, SPLIT and JOIN

- **Split**, could be described as condition on the exit for a task, where more than one road emerge from the task. There are three kinds of splits: AND-split, OR-split, and XOR-split.
  - **AND-split**: The AND-Split is used to start a number of task instances simultaneously. It can be viewed as a specialization of the OR-Split, where work will be triggered to start on all outgoing flows.
  - **OR-split**: The OR-Split is used to trigger some, but not necessarily all outgoing flows to other tasks. It is best used when we won't know until run-time exactly what concurrent resultant work can lead from the completion of a task.
  - **XOR-split**: The XOR-Split is used to trigger only one outgoing flow. It is best used for automatically choosing between a number of possible exclusive alternatives once a task completes.

- **Join,** is a condition for the entrance to a task, where more than one link merges in to the same task. Also four kind types: Simple-Join, AND-join, OR-join, and XOR-join.
  - **AND-join**: A task with an AND-Join will wait to receive completed work from all of its incoming flows before beginning. It is typically used to synchronise pre-requisite activities that must be completed before some new piece of work may begin.
  - **OR-join**: The OR-Join ensures that a task waits until all incoming flows have either finished, or will never finish. OR-Joins are "smart": they will only wait for something if it is necessary to wait. However, understanding models with OR-joins can be tricky and therefore OR-joins should be used sparingly.
  - **XOR-join**: Once any work has completed on an incoming flow, a task with an XOR-Join will be capable of beginning work. It is typically used to allow new work to start so long as one of several different pieces of earlier work have been completed.

There are other operators as Composition, Multiple Instance, Cancellation, etc. to cover all the identified workflow patterns with features as Synchronizations, Choices, Multiple instances, Exception handling.

Resource patterns deal with the entities capable of doing work. It is the most powerful process specification language for capturing resourcing requirements.

Data Data-flow patterns are focusing on the definition and ways to handle data, The data can be updated between different activities and it is necessary to handle in correct way and have the control on this. YAWL uses XML Schema, XPath and XQuery to do that. Provides a set of primitive data types and the ability to define user defined data types (XML Schema) and also supports variables, input/output parameters, and run-time variable transformations (XPath and XQuery).

About **Exception handling patterns** deal with the various exception cases and their handling.

4.2.1 Conclusions

While not being standardized, YAWL looks like the most complete workflow language to be accepted by the industry and by research environments. Currently cover more than 200 workflow patterns and offer an integrated framework to contemplate all the phases related to a workflow in business process, design graphically the workflow of the business process, generate the output xml file to import inside the execution engine and finally the execution and verification of these.

4.3 BPEL 2.0

The Business Process Execution Language (BPEL) [18,19] also know WS-BPEL, is an XML-based language for describing business processes and is based on web services orchestration allowing the composition of the web services with a process oriented approach to Service oriented architecture (SOA) which practically implies that role of BPEL is primarily for execution of web services in right order such that the technology implementation of the business processes is flexible and in alignment with the business goals. The last Specification is BPEL 2.0 released in April 2007.

The design of the BPEL is based on concepts derived from web services languages WSFL (Web Services Flow Language) that uses a directed graph approach developed by IBM and XLANG developed by Microsoft. Since BPEL is based upon Web services and hence supports the web services technology standards such as WSDL, SOAP, and UDDI.

The key objective of BPEL is to standardize the format of business process flow definition so companies can work together seamlessly using Web services, for that BPEL offers a great vocabulary for describing business processes.

As WS-BPEL business processes are expressed in XML, they are human-readable and can be used by any XML processing facilities, enabling them to be produced and consumed within the XML stack. At the same time uses and extends WSDL to both provide and consume Web services in an abstract way, using WSDL to define service interfaces.

BPEL describes a variety of XML elements to define business processes, such as:

- **Partners**: The actors in a business transaction
- **Containers**: The messages that need to be transmitted
- **Operations**: The type of Web services that are required
- **Port types**: The kinds of Web services connections that are required for operations

And supports two different types of business processes:

- **Executable processes**: Models the actual behavior of a participant role in a business interaction. An executable process follows the orchestration paradigm and can be executed by an orchestration engine.
- **Abstract processes**: they utilize process descriptions that specify the mutually visible message exchange behavior of each of the parties involved in the protocol, without revealing their internal behavior.

WS-BPEL provides the orchestration service layer for Service Oriented Architecture (SOA) and for that purpose BPEL connects with web services to define the semantics of e-business processes, interfaces, and workflows. Also enables best practices, patterns and training to be leveraged from a variety of vendors.

BPEL allow the design of the Application and business services process-agnostic and reusable. The business process assumes the management and coordination of state, freeing constituent services from a number of design constraints. Additionally, the business process logic is centralized in one location, as opposed to being distributed across and embedded within multiple services.

In WS-BPEL a business processes interact with services through Web services invocations, and are themselves externalized as Web services. This recursive composition enables a BPEL process to leverage the interoperability provided by the lower levels of the Web Services stack, such as WSDL, SOAP, and WS-Addressing. Another feature is that Uses XML Schemas type definitions for the data model.

Many deployments will have multiple orchestration platforms due to embedding in tools and applications, organizational purchases, etc. WS-BPEL provides a common standard which provides for interoperability between the different platforms and the processes that execute on them.

BPEL Arquitecture

The three core components of BPEL are the:

- **BPEL Designer**: Also known as the Graphical User Interface is used to define a business process that would be independent of the underlying applications. It is graphical and intuitive for the Business experts to define the process without being overly technical in depth. The output is a BPEL process flow logic template. All Web services needed for the specific business process would be included in the flow that uses the designer.

- **Process flow template**: The process flow format should be compliant to the BPEL specification. It captures the business process flow logic. which has been generated from the BPEL designer at design time and executed by the BPEL Engine at runtime.

- **BPEL Engine**: Basically is the runtime environment that executes any process flow template compatible to the BPEL standard. Functionality includes invocation of the Web services, mapping of the data content, error handling, transactionality, security, and so forth. Typically, the BPEL Engine would be integrated within the Application Server.

4.3.1 Tools

Different tools, both Open Source and Commercials has support to BPEL, in most cases for the design and for execution of business processes:

- **ActiveBPEL Designer and ActiveBPEL engine**

  ActiveBPEL Designer are a free, Eclipse-based BPEL authoring environment with rich functionality for BPEL process design, debugging, and simulation. Includes a built-in BPEL engine for desktop testing and deployments. ActiveBPLE engine is a runtime environment capable to execute process definitions created to the BPEL specifications.

  Source: http://www.activebpel.org

- **Sparx Systems upgrades Enterprise Architect 7.5**

  Commercial tool focused in UML modeling, but currently with support to the development of BPEL models by referencing imported WSDLs and XML Schema.

  Source: http://www.sparxsystems.com.au/products/ea/index.html

- **WebSphere Studio Application Developer Integration Edition v5.1**

It is a commercial tool at the top of the range application development environment that is needed to develop all the functionality covered by BPEL. It is built using Eclipse and has a consistent and powerful set of visual editors to define the process flow, message transformation and business rules.

Source: http://www-01.ibm.com/software/integration/wsadie/

4.3.2 Conclusions

Currently BPEL is the industry standard language for expressing business processes and is supported practically in all the BPM Systems products available in the market; the reason underneath is that WS-BPEL provides the standards-based platforms which reduce proprietary solutions and facilitate migration from one vendor platform to another. WS-BPEL processes will run on any WS-BPEL-compliant engine.

BPEL focuses exclusively on the executable aspects of the process, and does not contain elements to represent the graphical aspects of a process diagram, or human oriented processes.

5. Comparison and assessment of process modelling languages

Once presented different notations and languages for modeling and for business process definition, this section covers a comparison among them, taking into account a series of notation features that should belong to the field of software engineering to be able to be applied to the certification metamodel.

5.1 Comparison

In this section we perform the comparison of the selected modeling languages according to the criteria set out bellow. Regarding the different notations and languages, the comparison is sometimes hard because accurate description is often missing, not in all Process Models Notation or BPML's the metamodel is available and sometimes elements have ambiguous meanings. Nevertheless in accordance with the existing literature and the theory about business process modeling languages, we have identified a set of criteria relevant for the motivation of this analisys (Certification metamodel):

1. **Expressiveness**: Also know as Semantic Richness, reflects the expressive power of a process modeling language that is governed by its ability to express specific process requirements reflecting the purpose of process modeling and execution. A process model is required to be complete, which should contain structure, data, execution, temporal, and transactional information of the business process, for it is essential to give these support different notations for workflow patterns, the ability to represent roles and allocating these to different tasks.

2. **Graphical Representation**: This is closely related to the understandability of process models which is dependent on the possible users of a process model. Capability to represent and manage expressiveness features by means of graphical elements. Then multiple conceptual perspectives/views should be offered to be understood by different people who are not specialists in modeling.

3. **Abstraction and Modularization** : The modularity of modeling languages can be measured by considering the support for abstract processes and sub-processes. Abstraction prevents from revealing the underlying layers of a process, hence improving the understandability of process models for non-developer actors. Modular design reduces the complexity and it also increases

the ability to hide unnecessary information for different model users and various model uses.

4. **Variability and Reusability**: The ability to use previously modeled processes increases the speed and accuracy of modeling. It also reduces the time required for understanding the models. Also capabilities to specify repositories of processes that allow to manage the processes with features as monitoring, control and planning of these processes

5. **Executability**: A process modeling language may support to define operational models. Operational models are executable and easily enactable, this feature is referred to Enactment Support offered by the standard.

6. **Complexity Management**: The measures of the difficulty to model, analyze, and deploy a process model , as well as the support for the dynamic and changing business process.

7. **Quality**: Ability to specify the quality characteristics of business processes.

8. **Artifact description:** Ability to describe artifacts and their features.

9. **Adaptability**: Is the ability of the workflow processes to react to exceptional circumstances, which may or may not be foreseen, and generally would affect one or a few process instances, here the Exceptions are indispensable parts of business process-modeling. Exception handling features increase modeling difficulty but also increase the adaptability of models to exceptional circumstances. Improved adaptability helps model users to predict all model behavior in the time that exceptions occur.

10. **Availability and tool support**: A process modeling language should be available for the whole community as free as possible. This feature can include the process modeling language specifications documentation, and also specific computer tools to work with.

The following table contains a formal evaluation for the analyzed standards, in the columns, the standards are indicated and in the rows all the criteria defined before.

	SPEM 2.0	OPEN	BPDM	UML AD	BPMN 2.0	ARIS -EPC	IDEF3	XPDL 2.1	YAWL	BPEL
Expressiveness	*	+	+	+	+	*	+	+	+	+
Graphical Representation	+	+	-	+	+	+	+	+	+	+
Abstraction and Modularization	+	+	+	+	+	+	-	+	+	+
Variability and Reusability	+	+	-	+	-	-	-	-	-	-
Executability	+	+	+	+	+	+	+	+	+	+
Complexity Management	*	+	+	+	+	+	+	+	+	+
Quality	-	+	-	-	-	+	-	-	-	-
Artifact description	+	+	-	+	+	+	+	-	+	-
Adaptability	-	-	+	+	+	+	+	+	+	+
Availability and tool support	+	-	-	+	+	+	+	+	+	+

[+] - Supported **[-]** - Not Supported **[*]** - Supported by complementary notations

Currently there is a tendency to express the processes in two ways, first with a graphical notation that supports the most workflow patterns and secondly using a language expressed in XML that represents the described graphically and allow to exchange process definitions between different tools.

After analyzing the different standards and taken into account the individual conclusions, BPMN, XPDL y BPEL, each of them with its own aims and objectives could be the best option currently present in the market for modeling business processes.

BPMN 2.0 as graphical notation to describe the business process and workflow in order to be understandable by all users participants in the process, XPDL 2.1 as a format for storing and exchanging definitions of processes enabling a process in BPMN modeling tool for another tool to read and BPEL as an execution language specifying the web services technology as expressed in either of the above notations.

6. Conclusion

This deliverable analyses the most relevant process modeling notations and metamodels. The goal is to define a cohesive link between the eDIANA certification tool prototype and the specification of development processes in different languages. This link will be translated in extensions to the prototype as entry points that will connect the standard requirements with the process status, and the evidences required by standards.

The eDIANA Certification prototype is supported by a common metamodel to describe multiple standards and eDIANA-specific requirements and their evidences. The process models help to get a more cohesive certification approach by allowing developers to update development process status and artifacts with the certification requirements, and by providing a dynamic and reliable monitoring of the project against the selected standard. This will reduce certification efforts and costs.

Once analyzed the process modeling notations and standards, the most natural approach is to define a relationship between the Certification Requirements (central concept in the eDIANA Certification metamodel) and process elements such as tasks, activities, and input/output artifacts. The connection could be through a process management tool (for example Eclipse EMF or ESI's Process Factory) that would allow the definition of the business process model and its monitoring. Once a relevant activity for certification is executed, an event would be triggered from the process management tool in order to activate the evidence of the related Certification Requirement in the eDIANA certification tool.

The next Figure describes the possible links between the eDIANA certification tool and process management tools. With this approach, we intend to:

- Simplify certification assessment by tracking evidences from early phases. The link of Module A (see D6.3-B for full details) with a process management tool allows for identifying process activities and tasks with certification requirements.

- Automatically update the status of project process regarding certification requirements. The link of Module B with a process management tool allows for dynamically update process status with certification compliance actions.

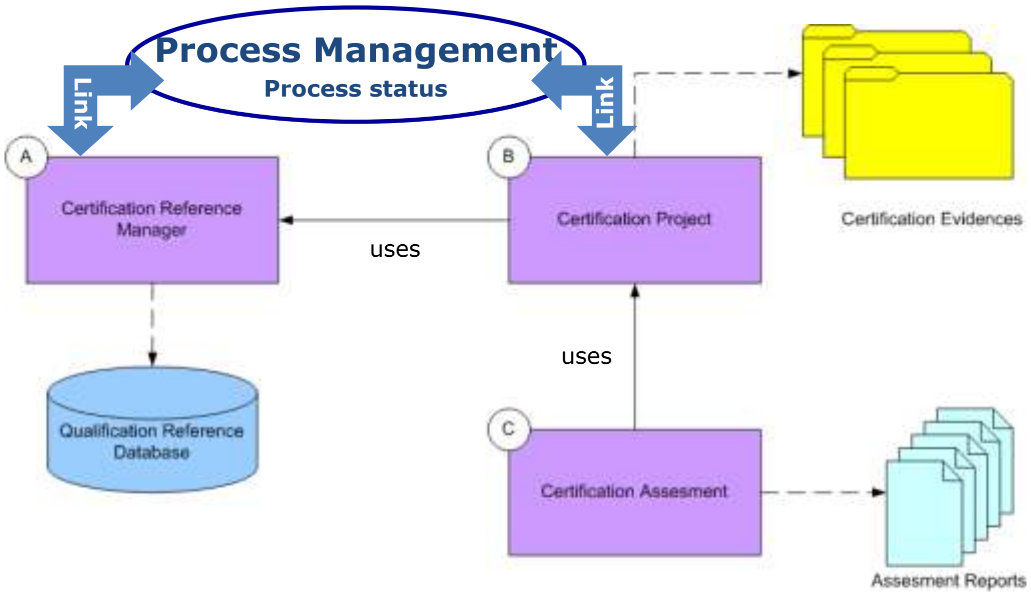


Figure 34: eDIANA Certification prototype (modules A, B and C) linked to a Process Management tool

The connection between module A and the process management tool might be done as follows. Certification requirements and the activities of the process model can be instantiated through the "Qualification Requirements" concept in the metamodel. Qualification Requirements can be standard requirements, a guideline, internal best practices, etc. On the other hand, a process management tool maintains an enacted description of a development project. By linking activities and tasks of this enacted process description with Qualification Requirements, we are providing a partial evidence of requirements accomplishment.

The connection between module B and the process management tool might be done as follows. As Project Requirements (managed by module B) are associated to Qualification Requirement, it would be easy associate evidences upadated in the process execution to the Project Requirement, allowing in this way a kind of automation between the processes and the evidences that will be use in the assessment. If the connection is made through the Project Requirements, the process could be ad-hoc to the company internal practices or rules, and the standard could be left more generic.

Acknowledgements

The eDIANA Consortium would like to acknowledge the financial support of the European Commission and National Public Authorities from Spain, Netherlands, Germany, Finland and Italy under the ARTEMIS Joint Technology Initiative.

References

[1] Osterwalder, A.; Pigneur, Y.; Tucci, C.L.: Clarifying Business Models: Origins, Present, and Future of the Concept, Communications of AIS, 2006, pp17

[2] Karagiannis, D.; Kühn, H.: Metamodelling Platforms, in: Bauknecht, K.; Min Tjoa, A.; Quirchmayer, G. (Eds.): Proceedings of the Third International Conference EC-Web 2002, 2002, pp182

[3] Kühn, H.; Bayer, F.; Junginger, S.; Karagiannis, D.:Enterprise Model Integration, Proceedings of the 4th International Conference ECWeb 2003, p3

[4] OASIS: Reference Model for Service Oriented Architecture 1.0, Committee Specification 1

[5] Workflow Management Coalition: Workflow Management Coalition, Terminology & Glossary, Document Number WFMC-TC-1011, Issues 3.0, February 1999, http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

[6] OMG - Object Management Group: http://www.omg.org/

[7] SPEM 2.0 Specification: http://www.omg.org/spec/SPEM/2.0/

[8] MOF 2.0 – Meta-Object Facility: http://www.omg.org/spec/MOF/2.0/

[9] BPDM Specification: http://www.omg.org/spec/BPDM/

[10] Wikipedia: Unified Modeling Language, http://en.wikipedia.org/wiki/Unified_Modeling_Language (October 2010)

[11] Object Management Group (OMG): Unified Modelling Language: Superstructure, version 2.1.1, 2007, p 221

[12] OMG: http://www.omg.org/spec/BPMN/2.0/

[13] Figure extracted from Poster BPMN 2.0 Notation: http://bpmb.de/posterParticipante

[14] IDS Scheer AG (Eds.): ARIS Platform, http://www.ids-scheer.com, 2010.

[15] IDEF – Integrated Definition Methods, http://www.idef.com/IDEF3.htm

[16] XPDL 2.1 Spec: http://www.wfmc.org/View-document-details/WFMC-TC-1025-Oct-10-08-A-Final-XPDL-2.1-Specification.html

[17] YAWL: http://www.yawlfoundation.org/

[18] BPEL 2.0: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[19] Wikipedia, Business Process Execution Language
http://en.wikipedia.org/wiki/Business_Process_Execution_Language