Workflow Management Coalition
Interface 1: Process Definition Interchange
Process Model

Document Number WfMC TC-1016-P

Document Status - 7.05 beta
(public distribution of beta)

Issued on August 5, 1998

Author:
Work Group 1

Send comments to: Workflow Management Coalition
WfMC@wfmc.org or
robert@metasoft.com

Remark:

This is the Beta release approved at the Stockholm meeting in July, 1998. It is distributed in order to allow discussion and feedback with the intention of preparing release 1.0 to be voted on at the October 1998 meeting in Toronto. Proposals for changes should be submitted to the Working Group One mailing list or robert@metasoft.com.

1. Introduction

1.1. Purpose

The WfMC has identified 5 functional interfaces to a workflow service as part of its standardisation programme. This specification forms part of the documentation relating to “Interface 1” - supporting Process Definition Import and Export. This interface includes a common meta-model for describing the process definition (this specification) and also a textual grammar for the interchange of process definitions (Workflow Process Definition Language - WPDL) and APIs for the manipulation of process definition data.

1.2. Audience

The intended audience of this document includes all participants in the workflow industry. Comments should be addressed to the Workflow Management Coalition.

1.3. Overview

A variety of different tools may be used to analyse, model, describe and document a business process. The workflow process definition interface defines a common interchange format, which supports the transfer of workflow process definitions between separate products.

The interface also defines a formal separation between the development and run-time environments, enabling a process definition, generated by one modelling tool, to be used as input to a number of different workflow run-time products. This meets the often expressed user’s requirements for the independence of modelling and workflow run-time products.

A workflow process definition, generated by a build-time tool, is capable of interpretation in different workflow run-time products. Process definitions transferred between these products or stored in a separate repository are accessible via that common interchange format.

Independent of the transfer mechanism itself (batch or API based), this standardised format describes a formal documentation of a workflow process, which focuses the information content of build-time definitions.

To provide a common method to access and describe workflow definitions, a workflow process definition meta-data model has been established. This meta-data model identifies commonly used entities within a process definition. A variety of attributes describe the characteristics of this limited set of entities. Based on this model, vendor specific tools can transfer models via a common exchange format.

The transfer mechanism could be either API-based or batch oriented (via files or memory transfers).

One of the key elements of the WPDL is its extensibility to handle information used by a variety of different tools. The WPDL may never be capable of supporting all additional information

requirements in all tools. Based upon a limited number of entities that describe a workflow process definition (the "Minimum Meta Model"), the WPDL supports a number of differing approaches.

One of the most important elements of the WPDL is a generic construct which supports vendor specific attributes for use within the common representation. We recommend that any missing attributes be proposed to the WfMC interface/1 workgroup for inclusion in future releases.

Both batch and API based interface architectures may implement a "filter" mechanism to support the information exchange. Via this "WfMC interface/1 layer" the input/output of build-time information can be translated to the standardised WPDL or to the vendor specific representation.

Furthermore the WfMC layer can be located on the client application (i.e. if it is just a Windows-only modelling tool), or on the server side (see Figure 2-1). The effort for this standardised WfMC interface can be reduced to a minimum.

There are two possible approaches to the interchange of a workflow process definition:

A) Define build-time APIs for the creation of the objects, their attributes and relationships participating in a workflow.

B) Define a common language for describing workflow processes which can be transferred as part of a textual file.

This document describes the meta-model which is used to define the objects and attributes contained within a process definition. The WPDL grammar is directly related to these objects and attributes. This approach needs two interfaces to be provided by a vendor:

1. Import a workflow definition from a character stream of definitions according to the common process definition language into the vendor's internal representation.
2. Export a workflow definition from the vendor's internal representation to a character stream according to the common process definition language.

All keywords used within the WPDL are based upon the WfMC Glossary.

1.4. Conformance

A vendor can not claim conformance to this or any other WfMC specification unless specifically authorised to make that claim by the WfMC. WfMC grants this permission only upon the verification of the particular vendor’s implementation of the published specification, according to applicable test procedures defined by WfMC.

A conforming implementation of this Functional Area of the Workflow Management Coalition specification includes the implementation of the relevant portions of the other functional areas: Client Application, Tool Invocation, Interoperability, Administrating and Monitoring.

Conformace for process definition import / export is essentially based upon conformance to the WPDL grammar and/or the API conformance class within the WAPI (Workflow API) definition. However, there is a mandatory minimum set of objects and attributes, as specified within this document, which must be supported within WPDL or available for manipulation by the process definition manipulation APIs. But: given the wide variation of capabilities in modelling tools, it is reasonable to assume that an individual tool might conform to the Interface 1 specification but not be able to swap complete definitions with all other conforming products. There is a two-level view of conformance:

1. Syntax, where on output the tool must generate valid, syntactically correct WPDL; on input, the tool must be able to read all valid WPDL. In this case, the translator should flag those expressions not applicable, and create appropriate descriptions so the modeller on the import side understands the nature and intent of the untranslated expressions.
2. Structure, where there is a mandatory set of objects and attributes. The suggestion is, to define a minimum set of objects and attributes needed to create a functioning model.

1.5. References

The following documents are associated with this document and should be used as a reference.

General background information:

- WfMC Terminology & Glossary (WfMC-TC-1011)
- WfMC Reference Model (WfMC-TC-1003)

WfMC API specifications, which include process definition manipulation APIs:

- WfMC Client Application API Specifications (WAPI) (WfMC-TC-1009)

Workflow process interoperability, used to support process invocation on a remote workflow service:

- Workflow Interoperability - Abstract Specifications (WfMC-TC-1012)
- Interoperability - Internet E-mail MIME Binding (WfMC-TC-1018)

There are two accompanying documents:

- The Organisational Model: (WfMC TC-1016-O) (Currently being revised as Resource Model)
- The representative business example (WfMC TC-1016-X) explaining the use of WPDL.

2. Overview of Process Definition Interchange

2.1. Introduction

A Process Definition is defined as:

The representation of a business process in a form which supports automated manipulation, such as modelling, or enactment by a workflow management system. The process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data, etc. (WfMC Glossary - WfMC-TC-1011)

It is the process definition which is interpreted by the workflow engine, acting as a template for the creation and control of instances of that process during process enactment. The process definition may contain references to sub-processes, separately defined, which make up part of the overall process definition. A loose distinction is sometimes drawn between production workflow, in which most of the procedural rules (i.e. elements of the process definition) are defined in advance, and ad-hoc workflow, in which the procedural rules may be created or modified during the operation of the process.

The abilities to create, interchange and modify a process definition are thus central to the operation of all classes of workflow system.

An initial process definition will contain at least the minimal set of objects and attributes necessary to initiate and support process execution. Some of these objects and attributes will be inherited by each created instance of the process. APIs are provided within WAPI *(WfMC Workflow Application Programming Interface -See WfMC-TC-1009)* to support operations on the attributes of such process instance entities. A further set of APIs is provided within WAPI for the manipulation of process definition objects and their attributes (for example to allow an authorised user to retrieve or modify parts of a process definition during enactment).

The WfMC Glossary also contains descriptions of, and common terminology for, the basic concepts embodied within a process definition such as activities, transitions, workflow relevant data and participants, etc.

2.2. Process Definition Interchange

The requirement to transfer process definitions, in whole or part, may occur for a number of business reasons:

- the use of a separate tool to define, model, analyse, simulate or document a business process prior to its enactment on a (different) workflow execution service (or services) enables separate selection of workflow and process definition/modelling tools, allowing the optimum product to be used for each part of the overall system
- the transfer and retrieval of a process definition to/from a common design repository, accessed by a number of different tools or run-time systems, may be desirable to provide a single formal point of responsibility for control of process definitions.
- the modification of an existing process definition by an authorised user may be required, during enactment, either on a one-off or persistent basis.
- the transfer of a process definition (in whole or part) from one workflow engine to another may be necessary to facilitate interoperability of that process between two or more workflow engines during process enactment. Such transfer may take place prior to, or in some cases during, enactment.

The workflow process definition interface defines a common interchange format, which supports the transfer of workflow process definitions between different products to satisfy the above scenarios.

The interface also defines a formal separation between the development and run-time environments, enabling a process definition, generated by one modelling tool, to be used as input to a number of different workflow run-time products. This meets the often expressed user's requirements for the independence of modelling and workflow run-time products.

A workflow process definition, generated by a build-time tool, is capable of interpretation in different workflow run-time products. Process definitions transferred between these products or stored in a separate repository are accessible via that common interchange format.

2.3. Approaches to Process Definition Interchange

A variety of different mechanisms may be required to transfer process definition data between systems according to the characteristics of the various business scenarios. In all cases the process definition must be expressed in a consistent form, which is derived from the common set of objects, relationships and attributes expressing its underlying concepts.

It is not the intent of the WfMC to place any arbitrary constraints on the forms of expression of the process definition. Current work is based upon representation:

1. As a process definition grammar, using lexical expressions to define constituent elements.

   This is intended for use principally within file based operations, enabling the transfer of a complete batch of process definition data, typically comprising one or more complete process definitions. When used in this way any appropriate file transfer service supporting ASCII level text interchange can be used to support transfer between products. The WPDL forms a common interchange standard which enables products to continue to support arbitrary internal

representations of process definitions with an import/export function to map to/from the standard at the product boundary.

2. As an object relationship model

   This is currently used to support API operations to query, retrieve or modify individual elements of the process definition. Programming bindings are provided for ”C” and ”IDL”. In this case a common underlying communications interface must be provided between the relevant products; typically based upon RPC and ORB mechanisms. Although it is possible to exchange complete process definitions by the repeated use of such APIs between compatible products, it is expected that their primary use will be for the selective retrieval and manipulation of parts of a process definition during process build or enactment. The transfer of complete (or large components of) process definitions may also be feasible using common object services such as relationship services and object repositories.

Future work is possible on other representations such as programmatic structures, derived from the common underlying concepts.

The principles of process definition interchange are illustrated in Figure 2-1

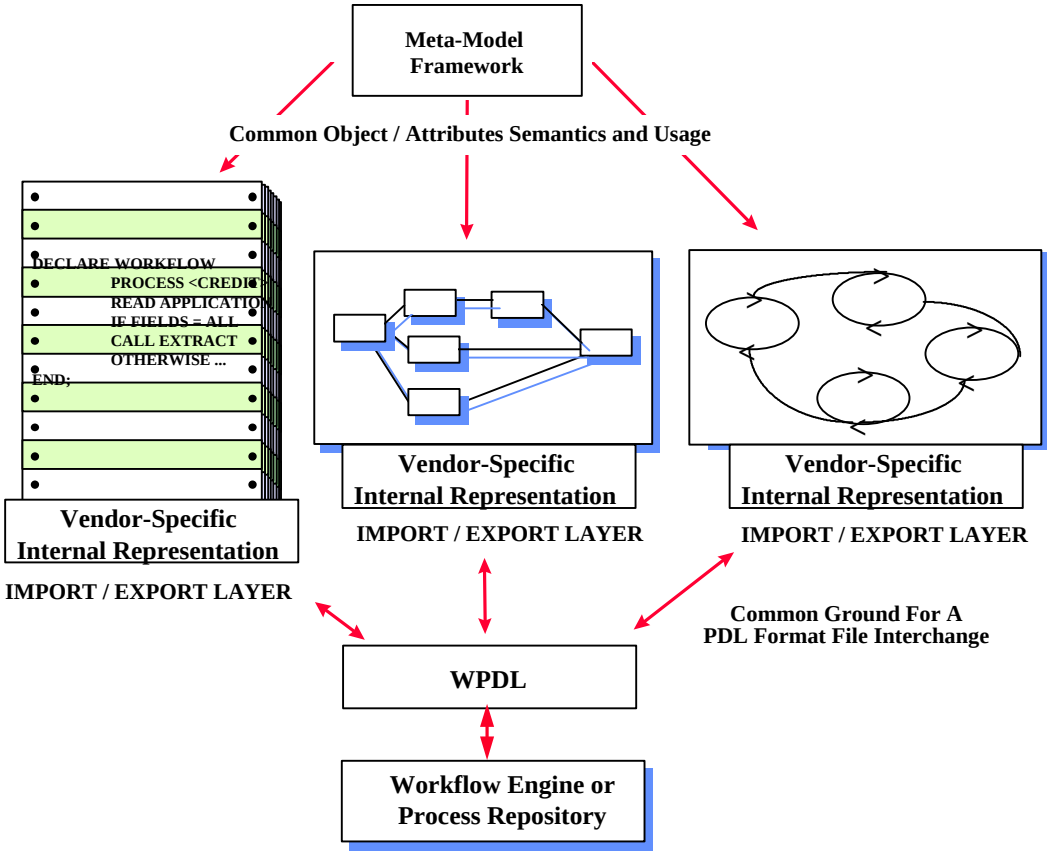

Figure 2-1: The Concept of the Process Definition Interchange

2.4. Specifications provided within this document

The WfMC specifications for Process Definition Import /Export comprise:

1. The process definition meta-model which describes the core objects within the process definition, their relationships and attributes. This “minimum meta-data model” identifies those commonly used entities within a process definition and describes their usage semantics. These attributes are expected to be relatively widely applicable to different products and workflow applications. This is provided within this document.
2. The Workflow Process Definition Language (WPDL) provides a formal language for the definition and exchange of a process definition using the objects and attributes defined within the meta-model. This is specified as Process Definition Import/Export - WPDL.
3. APIs within WAPI to provide manipulation of process definition entity attributes. These are defined within WfMC Client Application API Specifications (WAPI) (WfMC-TC-1009).

This separation into meta-model, WPDL and APIs allows for the possibility of exchanging process definition information by other techniques, when appropriate, in the future.

One of the key elements of the WPDL is its extensibility to handle information used by a variety of different tools, beyond the standardised data structures consistent with the meta-model. Due to the inherent variety within such tools, it is recognised that WPDL may never be capable of supporting or even predefining all potential additional information requirements in all tools. The "Minimum Meta Model" identifies an extensible set of objects / attributes sufficient to support common process definition characteristics. A small subset of the meta-model consists of mandatory elements; the remainder comprises optional, but common, elements.

Extensibility is provided by the facility to encompass additional object attributes (”extended attributes”) which can be included as extensions to the basic meta-model to meet the specific needs of an individual product or workflow system. Such ”extended attributes” can be added to the core meta-model over time in a standardised manner. WPDL includes a variety of data structures which may be used within such extended attributes.

Both batch and API based interface architectures may implement a "filter" mechanism to support the information exchange. All workflow or process modelling tools have an internal representation of process definition data which can be translated to a common interchange format on export, or translated from a common interchange format on import.

This approach needs two interfaces to be provided by a vendor:

1. Import a workflow definition from a character stream of definitions according to the common process definition language into the vendor's internal representation.
2. Export a workflow definition from the vendor's internal representation to a character stream according to the common process definition language.

This is identified in Figure 2-1 as an import/export function. The interface layer can be located on the client application (i.e. if it is just a Windows-only modelling tool), or on the server side (see

Figure 2-1), according to the style of products involved. The effort for this standardised WfMC interface can be reduced to a minimum.

2.5. Minimal State Model

This document primarily defines the Workflow Process Definition Interface in a descriptive way. However, since a build time representation of the process definition will subsequently be used to support enactment it is necessary to define a number of basic principles and semantics of the execution time environment.

The following minimal state model identifies the basic states of a process instance (for a more complete description see Ch.6.1):

- *notStarted* - the Process Instance has been created, but has not started yet.
- *running* - the Process Instance is executing.
- *suspended* - execution of the Process Instance is temporarily suspended.
- *completed* - enactment of the Process Instance has been finished and it has completed normally.

For most parts of this document it is assumed that the Process Instance is in the *running* state. We refer to an execution not changing this state as a *normal execution*.

To describe the semantics for those parts relating to the start and end of a Process Instance execution the transitions from *notStarted* to *running* and from *running* to *completed* will be referred to, respectively as the *START_TRANSITION* and the *END_TRANSITION*.

To restrict the scope of the validity of the semantics defined in this document sometimes a reference to a transition from *running* to *suspended* is required. When a reference to this transition is made it describes under which conditions this transition will take place. For the transition the only assumption is that the normal processing does not continue and the actual workflow participant, who is responsible for this workflow Process Instance is sent a notice. This document neither describes the notice itself nor makes any assumption about the possible subsequent operations, nor does it describe the way of detecting that this transition will be performed. This transition will be referred to as the *ERROR_SUSPEND* transition.

Examples for the notice to the Responsible are e.g. an E-mail or the Responsible sees the Process Instance in the Worklist with appropriate information. An example of the use of the *ERROR_SUSPEND* transition is that the engine has detected that there is no possibility for a transition following the completion of an Activity Instance, but that Activity Instance is not permitted to be the last one in a Process Instance during normal execution.

3. Meta-Model and Informal Description

3.1. Overview

The Meta-Model describes the top level entities contained within a Workflow Process Definition, their relationships and attributes (including some which may be defined for simulation purposes rather than workflow 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.

The top level entities are shown in the following figure:

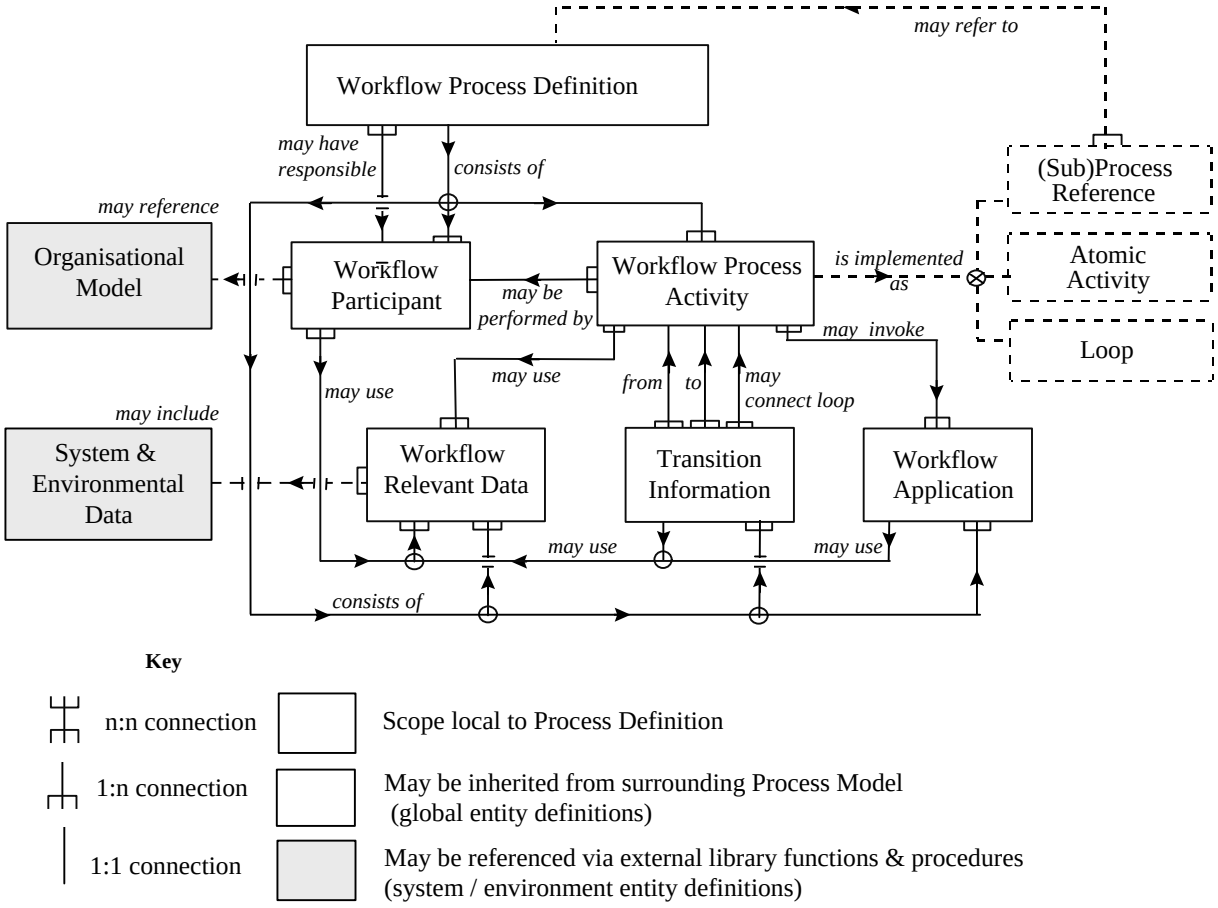

Figure 3-1: ***Meta-Model top level entities***

For each of the above entities, there is an associated set of attributes (some mandatory and others optional) which describe the characteristics of the entity. The following sections describe these entity / attributes in more detail. Where there is a need to use additional characteristics to those defined in this specification, “extended attributes” for various entities may be user defined to allow

extension of the scope of the meta-model in a controlled manner. The WfMC will review potential usage of such extended attributes on a periodic basis, with a view to incorporating them as standard attributes where appropriate.

3.1.1. Entities Overview

The meta-model identifies the basic set of entities and attributes used in the exchange of process definitions. The top level entities are as follows:

3.1.1.1.Workflow Process Definition

This describes the process itself, i.e. ID and textual description, and provides other optional information associated with administration of the process definition (creation date, author, etc) or to be used at process level during process execution (initiation parameters to be used, execution priority, time limits to be checked, person to be notified, simulation attributes, etc) . The Workflow Process Definition entity thus provides header information for the process definition and is therefore related to all other entities in that process.

3.1.1.2.Workflow Process Activity

A process definition consists of one or more activities, each comprising a logical, self-contained unit of work within the process definition. An activity represents work which will be processed by a combination of (organisational or system) resources (specified by participant assignment) and/or computer applications (specified by application assignment). Other optional information may be associated with the activity such as information on whether it is to be started / finished automatically by the workflow management system or its priority relative to other activities where contention for resource or system services occurs. Usage of specific workflow relevant data items by the activity may also be specified. The scope of an activity is local to a specific process definition (although see the description of a sub-flow activity below).

An activity may be **atomic** (the normal case) and in this case is the smallest unit of self contained work which is specified within the process (although an activity may generate several individual work items for presentation to a user invoking, for example, different IT tools).

An activity may be a sub-flow - in this case it is a container for the execution of a (separately specified) process definition, which may be executed locally within the same workflow service, or (using the process interoperability interface) on a remote service. The process definition identified within the sub-flow contains its own definition of activities, internal transitions, resource and application assignments (although these may be inherited from a common source). In- and out-parameters permit the exchange of any necessary workflow relevant data between calling and called process (and, where necessary, on return).

An activity may also be specified as a loop, which acts as controlling activity for repeated execution of a set of activities within the same process definition. In this case the set of looping activities is connected to the controlling (loop) activity by special loop begin/end transitions.

A number of activities may form an inline block, which is specified by particular transition

constructs, and may be used, for example, to represent an explosion of a higher level activity within a process definition as an alternative to a subflow.

Finally, a dummy activity is a skeletal activity which performs no work processing (and therefore has no associated resource or applications), but simply supports routing decisions within the incoming transitions and/or within the outgoing transitions.

3.1.1.3.Transition Information

Activities are related to one another via flow control conditions (transition information). Each individual transition has three elementary properties, the from-activity, the to-activity and the condition under which the transition is made. Transition from one activity to another may be conditional (involving expressions which are evaluated to permit or inhibit the transition) or unconditional. The transitions within a process may result in the sequential or parallel operation of individual activities within the process. The information related to associated split or join conditions is defined within the appropriate activity, split as a form of “post activity” processing in the from-activity, join as a form of “pre-activity” processing in the to- activity. (This approach allows the workflow control processing associated with process instance thread splitting and synchronisation to be managed as part of the associated activity, and retains transitions as simple route assignment functions.) The scope of a particular transition is local to the process definition which contains it and the associated activities.

More complex transitions which cannot be expressed using the simple elementary transition (ROUTE) attributes and the split and join functions associated with the from- and to- activities are formed using dummy activities, which can be specified as intermediate steps between real activities allowing additional combinations of split and/or join operations. Using the basic transition entity plus dummy activities, routing structures of arbitrary complexity can be specified. Since several different approaches to transition control exist within the industry, several conformance classes are specified within WPDL. These are described later in the document.

3.1.1.4.Workflow Participant Declaration

This provides descriptions of resources which can act as the performers of the various activities in the process definition. The particular resources which can be assigned to perform a particular activity are specified by an attribute of the activity, participant assignment, which links the activity to the set of resources (within the workflow participant declaration) which may be allocated to it. The workflow participant declaration does not necessarily refer to a single person, but may also identify a set of people of appropriate skill or responsibility, or machine automata resources rather than human. The meta-model includes four simple types of resources which may be defined within the workflow participant declaration.

3.1.1.5.Organisational Model

In more sophisticated scenarios the participant declaration may refer to an Organisational Model, external to the workflow process definitions, which enables the evaluation of more complex expressions, including reference to business functions and organisational entities and relationships. Reference to separate resource assignment expressions may also be required under various circumstances. This first version of specification does not include a fully standardised specification for either OM or process history functions; such functions can be realised by use of the extended library functions.

3.1.1.6. Workflow Application Declaration

This provides descriptions of the IT applications which may be invoked by the workflow service to support, or wholly automate, the processing associated with each activity, and identified within the activity by an application assignment attribute (or attributes). Such applications may be generic industry tools, specific departmental or enterprise services, or localised procedures implemented within the framework of the workflow management system. The workflow application definition reflects the interface between the workflow engine and the application, including any parameters to be passed.

3.1.1.7. Workflow Relevant Data

This defines the data which is created and used within each process instance during process execution. The data is made available to activities or applications executed during the workflow and may be used to pass persistent information or intermediate results between activities and/or for evaluation in conditional expressions such as in transitions or participant assignment. Workflow relevant data is of a particular type; WPDL includes definition of various basic and complex data types, (including date, string, etc..) Activities, invoked applications and/or transition conditions may refer to workflow process relevant data.

3.1.1.8. System & Environmental Data

This is data which is maintained by the workflow management system or the local system environment, but which may be accessed by workflow activities or used by the workflow management system in the evaluation of conditional expressions in the same way as workflow relevant data. As such it may be regarded as an extension to workflow relevant data. A small number of standardised data entities (accessed by predefined library functions) are defined; others may be added through the extended library function mechanism.

3.1.1.9. Data Types and Expressions

The meta-model (and associated WPDL) assumes a number of standard data types (string, reference, integer, float, date/time, etc); such data types are relevant to workflow relevant data, system or environmental data or participant data. Expressions may be formed using such data types to support conditional evaluations.

The above entities contain the attributes which support a common description mechanism for process definitions and constitute the **Minimal Process Model**. This can be expanded, where necessary, through the use of extended attributes or extended library functions.

3.1.2. Process Definitions, Workflow Model & Process Repository

As indicated in the diagram above, the minimal process model includes various entities whose

scope may be wider than a single process definition. In particular the definitions of participants, applications and workflow relevant data may be referenced from a number of process definitions. The meta-model assumes the use of a common process definition repository, associated with the workflow management system, to hold the various entity types comprising the process definition. Within the repository itself and to support the efficient transfer of process definition data to/from the repository, the concept of a workflow model is introduced, which acts as a container for the grouping of common data entities from a number of different process definitions, to avoid redefinition within each individual process definition.

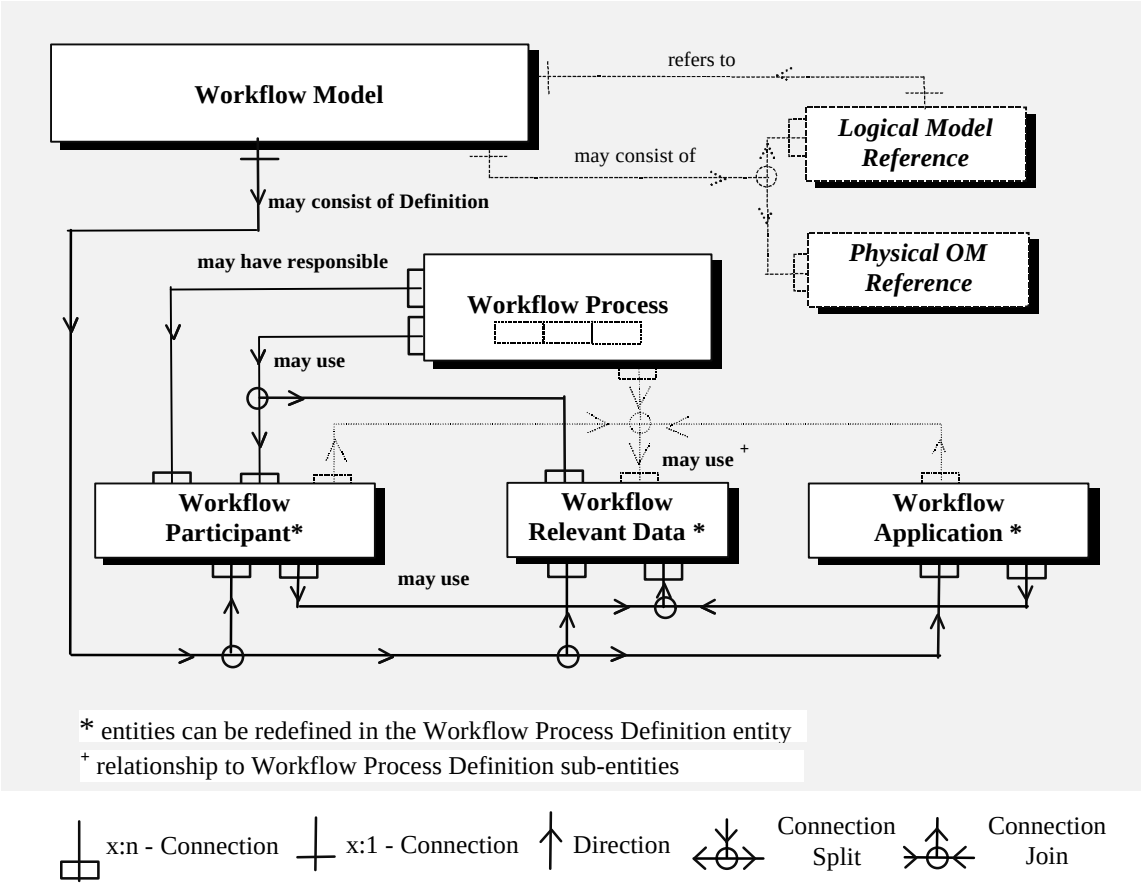

Figure 3-2: ***Workflow Process Model Entities***

The workflow model provides a container to hold a number of common attributes from the workflow process definition entity (author, version, status, etc). Each process definition contained within the process model will automatically inherit any common attributes from the process model container, unless they are separately re-specified locally within the process definition

Within a Process Model, the scope of the definitions of:

- workflow participant specification
- workflow application declaration, and
- workflow relevant data

are global and these entities can be referenced from all workflow process definitions (and associated activities and transitions) contained within the model. Restrictions on access to this (globally defined) workflow relevant data may be defined using a "RESTRICT_TO" attribute at the level of

process definition or process activity. This acts as a filter on the use of such workflow relevant data to support security or processing efficiency considerations during process enactment.

The process model may also provide a reference to two external object types - a Logical Model (separate instance of a Workflow Model) and/or a Physical Organisation Model.

The logical model reference allows the use within the process model or its contained objects of references to top level entities in the referenced external model:

- process ids for sub-flow reference
- workflow participant specifications
- workflow application declarations

Conventions on name and identifier management across different process models within the same repository address space to achieve any necessary global uniqueness are for user/vendor definition. The assumed convention during process enactment is that name reference searches follow the sequence:

- process ids - firstly within the same model (including any references to process definitions for remote execution on a different service), then within any externally referenced model
- applications / participants - firstly within the same model, then within any externally referenced model

Workflow relevant data naming must be unique within a process model; where such data is to passed between processes as parameters the convention at this version of specification is that copy semantics will be used. Responsibility rests with process designers / administrators to ensure consistent name / datatype usage within process definitions / models to support sub-flow operations (including any required remote process interoperability).

The overall structure of the process definition import/export interface and its relationship with the associated workflow service is shown in the diagram following.

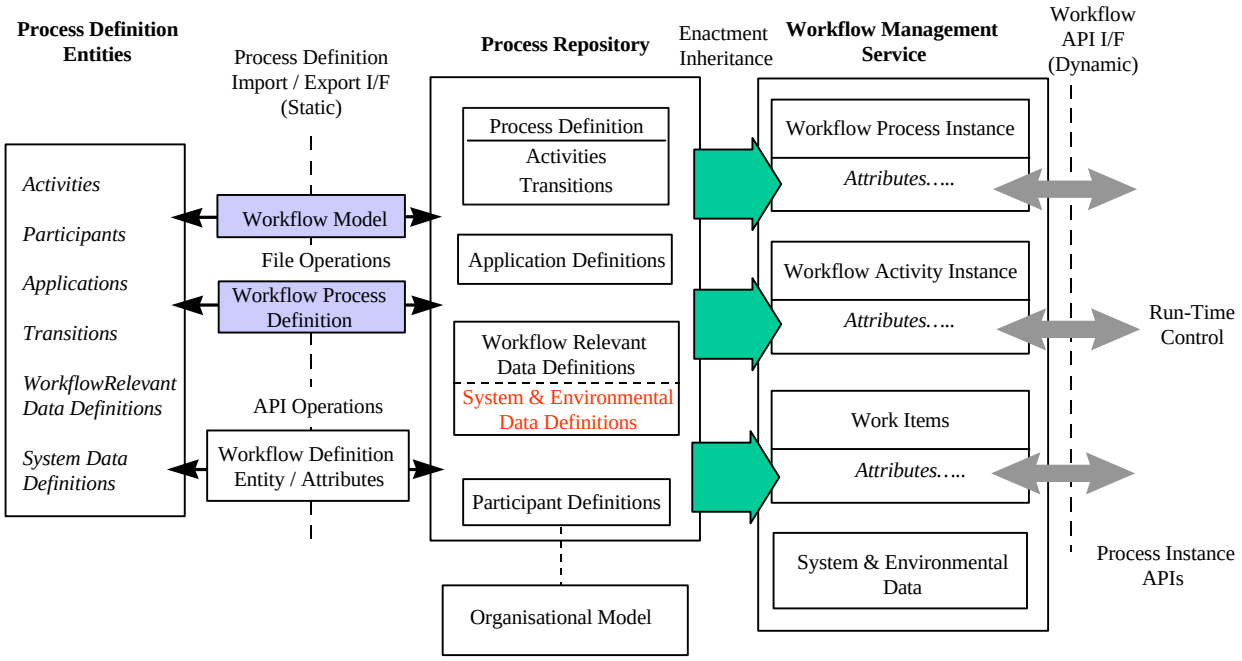

Figure 3-3: Process Definition Import/Export Interface

Further details on the conventions for use of data at the repository, model and process definition level are provided in Section 3.5 (*Workflow Model*).

3.1.3. Attributes Overview

The following table gives an overview of entities and attributes defined within WPDL. Bold typed attributes are mandatory, all others are optional. Italic typed attributes indicate relations to other entities. The triple dot "..." indicates the potential usage of extended attributes.

Workflow Model Definition	Workflow Process Definition	Workflow Process Activity	Transition Information	Workflow Application Declaration	Workflow Relevant Data	Workflow Participant Declaration
- **Id** - Name - Description	- **Id** - Name - Description	- **Id** - Name - Description	- **Id** - Name - Description	- **Id** - Name - Description	- **Id** - Name - Description	- **Id** - Name - Description
- **WPDL-Version** - **Source Vendor ID** - **Creation Date** - Version - Author - Characterset - Codepage - Country Key - Publication Status	- Creation Date - Version - Author - Characterset - Codepage - Country Key - Publication Status - Valid From Date - Valid To Date - Classification	- Automation Mode		- Tool Name	- **Type**	- **Type** - type related information

Workflow Model Definition	Workflow Process Definition	Workflow Process Activity	Transition Information	Workflow Application Declaration	Workflow Process Relevant Data	Workflow Participant Declaration
- Extended Library - Conformance Class	- Extended Library	- Split - Join - Loop - Inline Block				
	- Parameters		*- Condition*	*- Parameters*	*- Value*	
- Responsible *- External Model Ref* *- Access Restriction*	*- Responsible* *- Access Restriction*	*- Participant assignment* - Implementation *- Application assignment* *- Access Restriction*	***- From*** ***- To***			
- Documentation - Icon	- Documentation - Icon	- Documentation - Icon				
- Priority Unit - Cost Unit - Duration Unit -	- Priority - Duration - Cost - Waiting Time - Working Time	- Priority - Instantiation - Duration - Cost - Waiting Time - Working Time				- Strategy - Capacity - Cost - Prepare Time
...	...	...	...	...	...	...

Table 3-1: Overview of Entities and Attributes

The attributes can be divided into different groups.

- All entities have the attributes *id*, *name* and *description* in common.
- The second group contains specific attributes that characterise the respective entity.
- Parameters, conditions, and values refer to Workflow Process/Model Relevant Data and may be used in expressions.
- The fourth group contains attributes that reference other entities.
- Documentation and Icon attributes contain presentation information to be used by the executing engine.
- The sixth group contains information relevant for simulation and process optimisation (BPR-relevant information).
- For all entities extended attributes may be defined.

Further entities and predefined attributes may be added to the model to create future conformance levels. A short description and the semantics of all attributes is given in the subsequent chapters.

3.1.4. Name spaces

In the WPDL the following name spaces are distinguished,

- the name space for keywords and language specific constants (*token representations*),
- the name space for other tokens like strings, references etc.,
- the name spaces for identifiers of entities etc. (usually one per entity).

This has the benefit that the WPDL user needn't care about reserved words in the WPDL.

Inside these name spaces there is no structure given by the WPDL. It is therefore the responsibility of the user of the WPDL to organise their use.

3.1.5. Vendor or User specific Extensions

Although the meta-model and associated WPDL contain most of the constructs which are likely to be required in the exchange of process definitions, there may be circumstances under which additional information (user or vendor specific) will need to be included within a process definition. Users and vendors are encouraged to work as far as possible within the standard entity / attribute sets; the mechanisms described below to support extension provide a standardised means of expressing the extension for interchange purposes but may require localised system adaptation to provide any associated runtime support during process enactment.

3.1.5.1.Extended Attributes

The primary method to support such extensions is by the use of Extended Attributes, as described in the next chapter. Extended attributes are those defined by the user or vendor, where necessary, to express any additional entity characteristics which need to be exchanged between systems. Any run-time semantics associated with the use of the extended attribute during process enactment are separately specified and require bilateral agreement between the exporter and the importing workflow service.

3.1.5.2.Library Functions & Procedures

Library functions are defined to support access to common system or workflow environmental data which may be required for use in conditional expressions or for access by Applications as an extension of workflow relevant data. Where access to such data is not provided as standard within the workflow / system environment, a library procedure may be specified to fulfill the same purpose. Additional (“extended”) functions may also be defined, where necessary, on a user / vendor basis, each with an associated library procedure reference through which the function is evaluated.

3.1.5.3.Data Type Coercions

The meta-model and associated WPDL define a number of standard data types and expressions involving data types. For standard arithmetic expressions explicit coercions are defined; in other cases, if mixed data types are included within expressions, no specific coercions are defined and any applicable evaluation rules must be bilaterally agreed to. The use of library functions provides one approach for the support of such coercions.

3.1.5.4.Extended parameter mapping

No specific details of the scheme for encoding and passing parameter data is defined within this specification. Where parameters are passed on remote subprocess invocation using the Workflow Interoperability Specification (I/F 4), specifications are provided for the mapping of such parameters (for example into I/F 4 MIME exchanges) using the Set _Process_Instance_Attributes operation within the concrete syntax specification for interoperability. Any local scheme for parameter mapping and encoding is vendor defined on a product by product basis and lies outside the scope of this specification.

3.1.5.5.Extended flow control

The meta-model and WPDL does not specify any details of flow control internal to an activity - that is in specifying the sequence of work items and/or invoked applications, tools or procedures which may be generated within the activity. (See Activity specification - Application attribute usage). An extended flow control attribute may be user defined offering two alternative invocation schemes in these circumstances - sequential or parallel. Any further options, for example including conditional logic, is for user definition as an extended attribute. The approach to process definition is to discourage such conditional logic at work item level by refinement of the containing activity into separate activities with their own flow control logic.

3.1.5.6.Extended Library

Vendor/user provided functions and procedures may be added using an extended library declaration. This allows access to data in the local workflow manager or underlying system environment. Extended library definitions are contained within a process definition or, where applicable to a group of process definitions, within a process model definition. Functions and procedures are assumed to be implemented locally within the executing workflow environment and the appropriate identifier is declared within the extended library definition.

3.2. Elements Common for Multiple Entities

3.2.1. Notational Conventions for Informal Attribute Description

The Attributes are described in a common table format. In the case of complex Attribute

descriptions further refinement tables with a dedicated structure are used.

In general in the common table format the first column lists an (informal) name for an Attribute. An asterisk refers to a further description of that attribute. The column ***M/O*** indicates if the attribute is optional (O) or mandatory (M). ***WPDL Keyword*** lists the WPDL keyword used to identify the attribute. For the entity identifier it is always the <keyword> for that entity and is paired by an END_<keyword> at the end of each entity instance. The column ***Data Type*** either lists the data type (including type IDENTIFIER) of the item following the keyword or signals that there is a predefined set of WPDL keywords for that value that are listed in a separate table. In addition, constructors like LIST OF may be used. In the latter case a further table describes these values. DESCRIPTION provides a short textual description of the Attribute and describes the Default value if available.

The rows of the tables provide sequencing information: A full line between rows indicates that the Attributes are in sequence (both possible). A dotted line indicates alternatives, i.e. only one of them may appear. The alternatives following the first one start with a vertical bar "|". Rows not separated are explanations and represent groupings. If there is a double line between rows, then they are only for explanatory purposes described in one table; no sequencing information is expressed for them in this table but is provided elsewhere.

For better readability the Attributes are sometimes grouped. For reduction of complexity of the tables also a refinement of tables is sometimes used. To allow compact description of alternatives in one table sometimes only columns more to the right are separated by full lines. In this case the sequence is local to that alternative.

Lines are bold only for readability purposes; there is no additional meaning associated with them.

3.2.2. Common Attributes

3.2.2.1.Extended Attributes

Informal Description

Extended Attributes can be used in all entities, in Library Functions and Procedures and in External Declarations.

Attributes

Attribute Name	M/O	WPDL Keyword	Data Type	Description
Identifier	M	EXTENDED_ATTRIBUTE	IDENTIFIER	Used to identify the Extended Attribute
Attribute type	M		IDENTIFIER	Datatype, valid types are: *simple and complex data types*

Attribute Name	M/O	WPDL Keyword	Data Type	Description
Attribute value	M		*(a value of appropriate type)*	Preassignment of data for run time, *an initial value or result from a function call of appropriate type*
Attribute description	O		STRING	Textual description of the attribute

Table 3-2: Extended Attributes

3.2.2.2.Formal Parameters

Informal Description

Formal parameters can be used as attributes in Workflow Process and Workflow Application entities and in Library Functions and Procedures. These are the invocation parameters.

Attributes

Attribute Name	M/O	WPDL Keyword	Data Type	Description
Parameters				Formal Parameters that are passed during invocation and return of control (e.g. of an invoked application).
	O	IN_PARAMETERS	List of IDENTIFIER	The Input Parameters, e.g. for the invoked application.
	O	OUT_PARAMETERS	List of IDENTIFIER	The Output Parameters, e.g. of the invoked application.

Table 3-3: *Formal Parameters*

Parameter passing semantics

The **parameter passing semantics** is defined as:

(a) Any read-only formal parameters (parameter in the list of IN_PARAMETERS but not in the list of OUT_PARAMETERS) are initialised by the value of the corresponding actual parameter in the call (an expression). This is pass-by-value semantics.

(b) Any read/write formal parameters (same parameter in the list of IN_PARAMETERS as well as in the list of OUT_PARAMETERS) are initialised by the value of the corresponding actual (passed) parameter, which must be the identifier of a workflow relevant data entity. On completion of the process, the value of the formal out_parameter is copied back to the

original actual parameter (which must be the identifier of a workflow relevant data entity). This is copy-restore semantics.

(c) Any write-only formal parameters (parameter in the list of OUT_PARAMETERS but not in in the list of IN_PARAMETERS) are initialised to zero (strings will be set to the empty string, complex data will have each element set to zero). On completion of the process, the value of the formal out_parameter is copied back to the original actual parameter (which must be the identifier of a workflow relevant data entity). This is zero-restore semantics.

Concurrency semantics

Copying and restoring of parameters are treated as atomic operations; to avoid access conflicts from concurrent operations on workflow relevant data within the process instance these operations are serialised. Between copy and restore of (c) no locking is assumed and the returned parameter value will overwrite the local value (of the particular workflow relevant data item) at the time of the return call.

Formal-actual parameter mapping

The mapping of actual to formal parameters during invocation is defined by a parameter map list. The actual parameters are mapped 1:1 to the formal parameters in sequence. Type compatibility is required within the definitions and may be enforced by the run-time workflow system. The effects of violation are locally defined and do not form part of this specification

3.3. Process Model

3.3.1. Meta-Model

The meta-model identifies the basic set of entities and attributes for the exchange of process definitions. For a Process Definition the following entities must be defined, either explicitly at the level of the process definition, or by inheritance directly or via cross reference from a surrounding process model:

- **Workflow Process Activity**
- **Transition Information**
- **Workflow Participant Specification (OM Specification)**
- **Workflow Application Declaration**
- **Workflow Relevant Data**

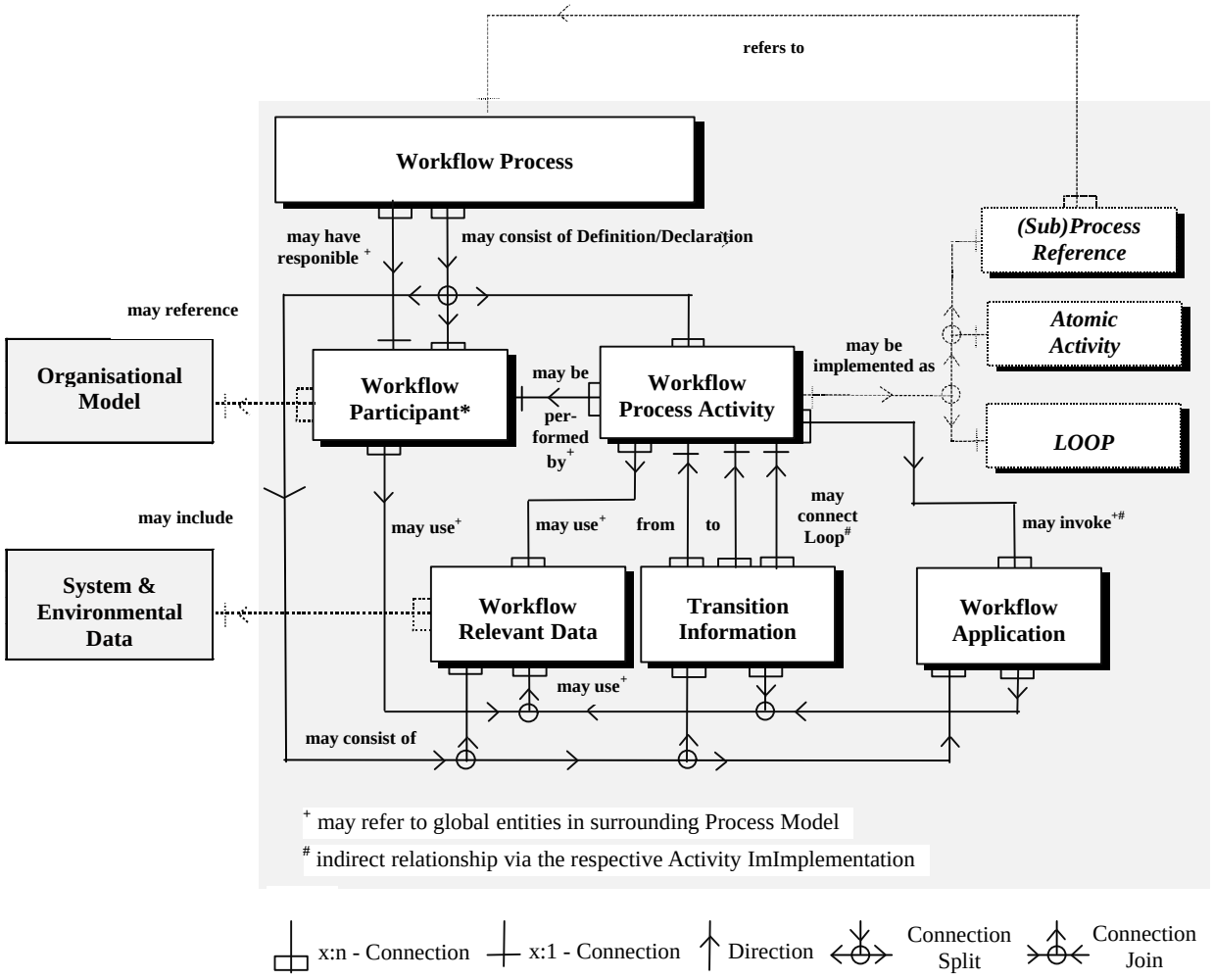

Figure 3-4: Workflow Process Definition Meta Model

These entities contain attributes which support a common description mechanism for processes. They are described in the subsequent document sections. Note that a Workflow Participant Specification is the either a Definition, or a Declaration using a minimal Organisational Model). In either case the link from an activity is by the participant assignment attribute

3.3.2. Workflow Process Definition

Informal Description

The Workflow Process Definition defines the elements that make up a workflow. It contains definitions or declarations, respectively, for Activity and, optionally, for Transition, Application, and Process Relevant Data entities. Attributes may be specified for administration relevant data like author, version, textual font and character set used, for runtime relevant data like priority, and for

BPR and simulation relevant data. In addition Extended Library elements may be defined.

A Workflow Process may run as a sub-process invoked as an implementation of an Activity of type Subflow; in this case parameters may be defined as attributes of the process.

Where a workflow process definition includes input parameters and is instanciated by means other than a subprocess call (for example by local event) the method for initialising any input parameters is locally defined. In such circumstances any workflow relevant data associated with the instanciated process definition which is included within the parameter list will be initialised to the value specified in the “default_value” (where specified). In such cases the value of the relevant process instance attributes would normally be set by WAPI call. Where workflow relevant data is not passed as an input parameter, initialised by WAPI call or initialised by “default_value” the result is undefined. Similarly where a subprocess terminates abnormally without returning out_parameter values to the calling process, the result is undefined. (Normally, for synchronous subprocess invocation this condition would be detected as a result of a “LIMIT” condition being exceeded allowing specific supervisory action.)

Scope and Name Space

In general the scope of the defined entity identifier and name is the surrounding entity. The identifier is unique in this scope. For the Process identifier and name the scope is the surrounding Workflow Model (chapter 3.5).

The name spaces of the defined entities are disjunct.

The Workflow Participant identifiers used in the RESPONSIBLE attribute have either to be declared in the surrounding Workflow Model or are inherited from a Model declared to be EXTERNAL.

3.3.2.1.Attributes

Attribute Name	M/O	WPDL Keyword	Data Type	Description
Process Identifier	M	WORKFLOW	IDENTIFIER	Used to identify the workflow process.
Process Name	O	NAME	STRING	Text Used to identify the workflow process.
Process Description	O	DESCRIPTION	STRING	Short textual description of the process.
Creation Date	O	CREATED	DATE	Creation date of workflow process definition.
Author	O	AUTHOR	STRING	Name of the author of this workflow process definition. (The one, who put it into the repository)
Version	O	VERSION	STRING	Version of this workflow process definition.

Attribute Name	M/O	WPDL Keyword	Data Type	Description
Character set	O	CHARACTERSET	STRING	Name of the character set. E.g. OEM for DOS, ANSI for Windows, ...
Codepage	O	CODEPAGE	STRING	The codepage used for the text parts. Default: Inherited from Model Definition.
Country key	O	COUNTRY_KEY	STRING	A country key. Default: Inherited from Model Definition.
Responsible*	O	RESPONSIBLE	PARTICIPANT *(expression)*	Workflow participant, who is responsible for this workflow process (usually an Organisational Unit or a Human). It is assumed that the responsible is the supervisor during execution of the process. In case of a state transition (see chapter 2.5) to *ERROR_SUSPEND* the responsible is notified. *(See also chapter 3.3.3.5)* Default: Inherited from Model Definition.
Publication Status*	O	STATUS	*Keyword*	Status of the Workflow Process Definition. Default: Inherited from Model Definition.
Valid From Date	O	VALID_FROM	DATE	The date that the workflow process definition is active from. Empty string means system date. Default: Inherited from Model Definition.
Valid To Date	O	VALID_TO	DATE	The date at which the process definition becomes valid. Empty string means unlimited validity. Default: Inherited from Model Definition.
Classification	O	CLASSIFICATION	STRING	Classification of process definition (i.e. *finance, sales, manufacturing etc*.).
Priority	O	PRIORITY	INTEGER	The priority of the process type. Default: Inherited from Model Definition.

Attribute Name	M/O	WPDL Keyword	Data Type	Description
Limit	O	LIMIT	INTEGER	Expected duration for time management purposes (e.g. starting an escalation procedure etc.) in units of DURATION_UNIT. It is counted from the starting date/time of the Process. The consequences of reaching the LIMIT value are not defined in this document (i.e. vendor specific). It is assumed that in this case at least the Responsible of the current process is notified of this situation.
Documentation	O	DOCUMENTATION	REFERENCE	Operating System specific path- and filename of help file/description file.
Icon	O	ICON	REFERENCE	Address (path- and filename) for an icon to represent the process definition.
Simulation Data				Estimations for simulation of a process
	O	DURATION	INTEGER	Expected duration time to perform a task in units of DURATION_UNIT.
	O	COST	STRING	Average cost for cost management purposes (used within analysis environment).
	O	WORKING_TIME	INTEGER	Describes the amount of time the performer of the activity needs to perform the task (time estimation) (working time is needed for analysis purposes and is provided by the evaluation of runtime parameters) in units of DURATION_UNIT.
	O	WAITING_TIME	INTEGER	Describes the amount of time which is needed to prepare the performance of the task (<time estimation>) (waiting time is provided by the analysis environment and may be updated by the runtime environment) in units of DURATION_UNIT.
Access Restriction	O	*RESTRICT_TO*	List of IDENTIFIER	List of Identifiers of Workflow Relevant Data defined in the surrounding Process Model Definition. Restricts access of globally defined Workflow Relevant Data to those listed. Default: No restriction.
Parameters*	O	*(see chapter 3.2.2.2)*		Parameters which are interchanged with the process (for use as Subprocess).
Extended library*	O	*(see chapter 3.5.5)*		Functions and procedures in an extended library *(for details see chapter 3.5.5)*

Table 3-4: Attributes of Entity Workflow Process

Publication Status

The Publication Status attribute describes the development status of this Workflow Process Definition. The predefined value keywords are defined; however, their semantics is left to the user.

Attribute Keyword	Value Keyword	Description
STATUS	UNDER_REVISION	Status of the Workflow Process Definition. (user defined semantics)
	| RELEASED	(user defined semantics)
	| UNDER_TEST	(user defined semantics)

Table 3-5: Entity Workflow Process: Values of Attribute Publication Status

3.3.3. Workflow Process Activity

Informal Description

The Workflow Activity Definition is used to define each elementary activity that makes up a workflow process. Attributes may be defined to specify Activity control information, implementation alternatives, Performer assignment, runtime relevant information like priority, and data used specifically in BPR and simulation situations (and not used within workflow enactment). In addition, restrictions on data access and to transition evaluation (e.g. Split and Join) can be described. Mandatory attributes are used to define the activity identifier and type; a small number of other attributes are optional but have common usage across all activity types. Other attribute usage depends upon the activity type as shown in the table below.

Scope

For the Activity identifier and name the scope is the surrounding Workflow Process (chapter 3.3.2).

Entity type relationships for different Activity types

The activity description is used to describe several different activity types. All these activities share the same (common) general activity attributes, but the usage of other attributes, particularly participant and application assignment and the use of workflow relevant data may be specialised to the activity type. The following table identifies the usage of other attributes / entity types for the different activity types.

Entity Types (usage within Activity Type)	**Activity Type**				
	Implementation Type				**Dummy (ROUTE)**
	None	**Application**	**Sub-flow**	**Loop**	
Transition Restriction	Normal	Normal	Normal, plus sub-flow call / return within activity	Normal, plus single loop control within activity	Normal; any additional controls implemented within Route activity
Participant Assignment	Normal	Normal	N/A	N/A	N/A
Application Assignment	None	Yes	N/A	N/A	N/A
Use of workflow Relevant Data	Normal	Normal	may be used in parameter passing	may be used in loop control conditions	may be used in routing control conditions

Notes on usage:

Transition restrictions, sub-flow, loop and route activities are described in the section on transitions.

In general, normal transition restrictions may be declared at the level of the activity boundary within the surrounding process, whereas specialised flow conditions (sub-flow, loop, or the internal part of a route activity) operate “internal” to the activity (but may reference activities within the surrounding process definition). The following diagram illustrates the generic structure of an activity and the above variants.

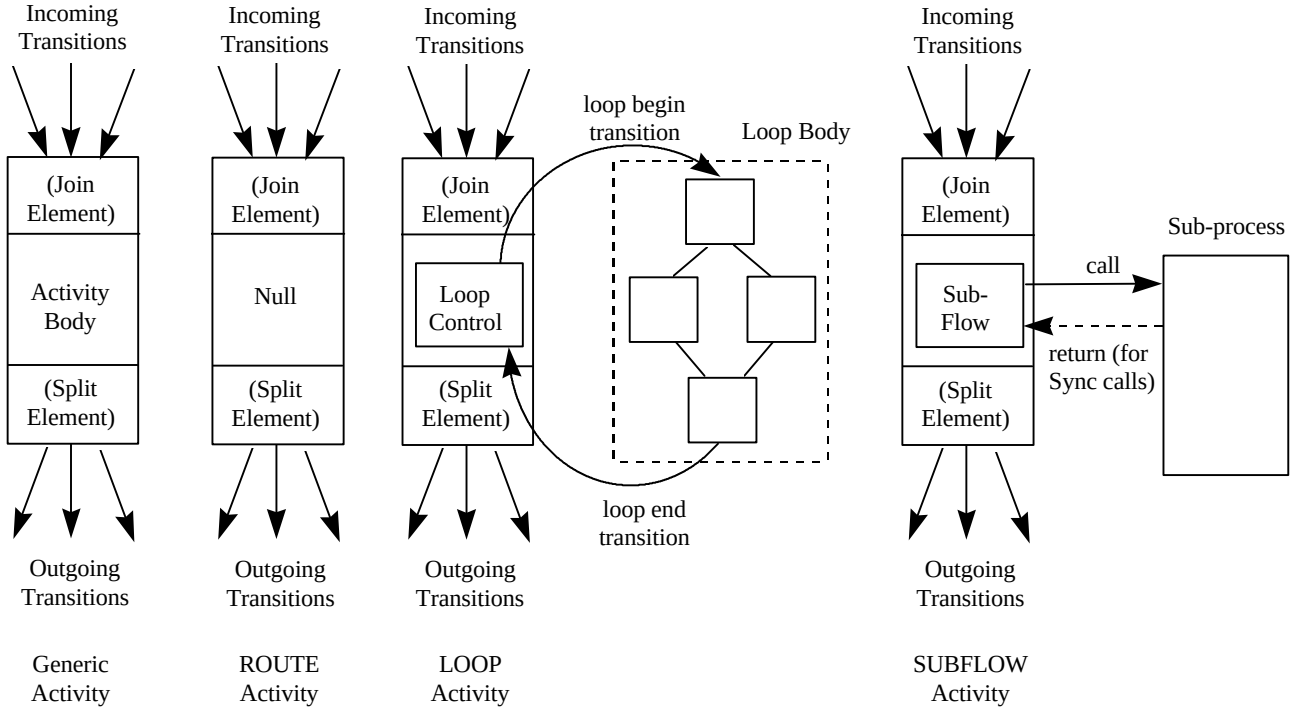

Figure 3-5: ***Activity Structures & Transition Conditions***

Where the implementation type is NONE, the workflow activity is manually controlled and its completion must be explicitly signaled to the workflow management system, for example using an appropriate API call. (Such activities might typically comprise instructions to the participant to undertake a non automated task of some type and inform a supervisor when completed.)

Application assignment is not relevant to sub-flow or manual activities; routing or loop controlling activities do not implement applications as such but may make reference to library procedures to invoke the routing or loop control logic.

Workflow relevant data may (potentially) be referenced within any activity although its use in manual activities is undefined through the process definition. Where an activity is of type sub-flow any in-parameters passed to the called (sub-) process must have been declared as workflow relevant data within the calling process / activity definition, or have been inherited from the surrounding process model. (Similar requirements apply to any out-parameters returned to the calling process.) Routing or loop controlling activities do not manipulate workflow relevant data directly, but may refer to such data within conditional expressions within the join/split/loop_control logic.

3.3.3.1.General Activity Attributes

Attribute Name	M/O	WPDL Keyword	Data Type	Description
Activity Identifier	M	ACTIVITY	IDENTIFIER	Used to identify the workflow process activity.
Activity Name	O	NAME	STRING	Text used to identify the workflow process activity.
Activity Description	O	DESCRIPTION	STRING	Short textual description of the activity.
Activity Kind Description	M	ROUTE		A "dummy" Activity
		| IMPLEMENTATION*	*(see below)*	A "regular" Activity *(details and further Attributes see below)*
Access Restriction	O	*RESTRICT_TO*	List of Identifier	List if Identifiers of Workflow Relevant Data. Restricts access to Workflow Relevant Data to those listed.
Transition Restrictions*	O	*(see below)*	*(see below)*	Provides further restrictions and context-related semantics description of Transitions *(details see below).*

Table 3-6: Attributes of Entity Workflow Process Activity

Route Activity kind

The ROUTE Activity is a "dummy" Activity that permits the expression of "cascading" Transition conditions (e.g. of the type "`IF condition-1 THEN TO Activity-1 ELSE IF condition-2 THEN TO Activity-2 ELSE Activity-3 ENDIF`"). Some vendors might implement "cascading" Transition conditions directly without requiring an Activity counterpart for a Route, others might require it. Wherever possible vendors and process designers are encouraged to structure such cascading conditions as an XOR split from the outgoing activity. Certain transition combinations cannot be expressed within a single transition list from the outgoing activity or a single incoming list to an activity . These cases require the use of one or more dummy activities; examples are:

- combination of XOR and AND split conditions on outgoing transitions from an activity
- combination of XOR and AND join conditions on incoming transitions to an activity
- transitions involving conditional AND joins of a subset of threads, with continuation of individual threads

A ROUTE Activity has neither a Performer nor an Application and its execution has no effect on Workflow Relevant Data or Application Data.

For simulation purposes the following Simulation Data values should be assumed: `DURATION` 0, `COST` "0", `WORKING_TIME` 0, `WAITING_TIME` 0. For `PRIORITY` and `INSTANTIATION` the maximum value should be assumed.

3.3.3.2.Activity Implementation Attributes

An Activity that is not a Route is a "regular" Activity and has an implementation and further Attributes.

Attribute Name	M/O	WPDL Keyword	Data Type	Description
Implementation alternative*	O	IMPLEMENTATION	*Keyword*	Mandatory if not a Route. Alternative implementations are “no”, “subflow” or “loop” (see below)
Participant Assignment*	O	PERFORMER	PERFORMER	Link to entity workflow participant. May be an expression. Default: Any Participant. (see below)
Automation Mode*				Execution control attribute: Description of the degree of automation of triggering and terminating an Activity.
	O	START_MODE	*Keyword*	Describes how the execution of an Activity is triggered.
	O	FINISH_MODE	*Keyword*	Describes how the system operates at the end of the Activity.
Priority	O	PRIORITY	INTEGER	A value that describes the initial priority of this activity when it starts execution. If this attribute is not defined but a priority is defined in the Process definition then that is used. By default it is assumed that the priority levels are the natural numbers starting with zero, and that the higher the value the higher the priority (i.e.: 0, 1, ...).
Documentation	O	DOCUMENTATION	REFERENCE	The address (e.g. path- and filename) for a help file or a description file of the activity.
Icon	O	ICON	REFERENCE	Address (path- and filename) for an icon to represent the activity.
Simulation Data				Estimations for simulation of an Activity. No default.
*	O	INSTANTIATION	*Keyword*	Defines the capability of an activity to be activated: *once* or *many times (multiple)*

Attribute Name	M/O	WPDL Keyword	Data Type	Description
	O	DURATION	INTEGER	Expected duration (summary of working time and waiting time) in units of DURATION_UNIT.
	O	COST	STRING	Average cost.
	O	WORKING_TIME	INTEGER	Average working time in units of DURATION_UNIT.
	O	WAITING_TIME	INTEGER	Average waiting time in units of DURATION_UNIT.

Table 3-7: Implementation Attributes of Entity Workflow Process Activity

3.3.3.3.Execution Control Attributes

These are attributes of an Activity that allow the definition of various activity-specific features for Activity execution control.

Automation Mode

This defines the degree of automation when triggering and terminating an Activity. There are two automation modes:

- ***Automatic mode*** is fully controlled by the Workflow engine, i.e. the engine proceeds with execution of the activity within the workflow automatically, as soon as any incoming transition conditions are satisfied. Similarly, completion of the activity and progression to any post activity conditional logic occurs automatically on termination of the final invoked application.
- ***Manual mode*** requires explicit user interaction to cause activity start or finish. In such systems the activity start and/or completion is as a result of explicit user action, normally via API call (WAPI) from a client application or tool agent.

The automation modes can be specified independently for the ***start*** and ***end*** of an Activity.

WPDL Keywords	Value Keywords	Description
START_MODE		Describes how the execution of an activity is triggered.
	AUTOMATIC	Triggered implicitly by the system. Default.
	| MANUAL	Triggered explicitly by the end user. Requires an appropriate API *(described in Interface 2).*
FINISH_MODE		Describes how the system operates at the end of the activity.
	AUTOMATIC	Implies an automatic return when the invoked application finishes control. Default.
	| MANUAL	The end user has to terminate the activity explicitly. Requires an appropriate API *(described in Interface 2).*

Table 3-8: Entity Workflow Process Activity - Automation Mode Attributes

3.3.3.4.Implementation Alternatives

An Activity may be implemented in one of four ways as described in the following table:

Implementation Alternative	Implementation Keyword	Description
No implementation	NO	Implementation by manual procedures(i.e. not supported by workflow)
Application	| APPLICATIONS	Implementation is supported by (one or more) application(s)
Subflow	| WORKFLOW	Implementation by a subprocess
Loop	| *LOOP*	Implementation is by a loop of other activities, connected by specific loop transitions.

Table 3-9: Implementation Alternatives of Entity Workflow Process Activity

It is assumed that the execution of the Activity is atomic with respect to the data under control of the Workflow engine. That implies that in the case of a system crash, an abort, or a cancellation of the Activity, the Workflow Relevant Data and the workflow control data are rolled back (automatically or by other means), or an appropriate compensating activity is applied. (This does not necessarily hold for audit data.) This version of the specification does not include any specific controls over data synchronisation or recovery (for example between workflow execution, subflows or applications under execution.)

No Implementation

No Implementation means that the implementation of this Activity is not supported by Workflow using automatically invoked applications or procedures. Two Alternatives have been identified as to how this may be used:

- It is a Manual Activity. In this case FINISH_MODE value MANUAL is required.
- It is an "implicit" activity, which is known to the Workflow Engine (e.g. by vendor-specific Extended Attributes) in terms of any processing requirements. An example is the Pre- and Post-processing Activities in a Workflow, which generate and clear hidden data when starting and terminating a process (e.g. managing the relationship to imaging system and archive). In this case the START_MODE and FINISH_MODE values AUTOMATIC are common.

(Note that application initiation may still be handled directly by the participant under local control in a manual activity; this lies outside the scope of the specification.)

Application

The Activity is implemented by (one or more) Generic Tools. A Generic Tool may be an application program (link to entity Workflow Application) or a (built-in or extended) Library Procedure. In the former case it is invoked via IF3 - see the Workflow Client Application API (WAPI - Interface 2); in the latter case a procedure is directly executed by the Workflow engine or

surrounding system environment.

Application kind Keyword	Data Type	Description
TOOL	IDENTIFIER	A generic tool identifier
| PROCEDURE	IDENTIFIER	A library procedure identifier

Table 3-10: Entity Workflow Process Activity - Implementation as Application

The Generic Tool may have parameters. The formal parameter descriptions, and thus the formal-actual parameter mappings, differ for Workflow Application and Library Functions (see chapter 3.3.5.2).

In case of implementation by two or more Generic Tools it is possible to provide an Extended Attribute containing an Extended Flow Control that describes the control flow between these Tools. The contents are vendor defined and may provide keywords for an execution sequence (e.g. "SEQUENTIAL" or "PARALLEL"). The Default is "SEQUENTIAL". (Note that the meta-model (or WPDL) does not support conditional logic for managing a control flow internal to an activity, for example to support alternative application invocation sequences or alternative performer assignments for different work-items within an activity. Where such logic is required, the activity may be refined to a series of activities of smaller granularity.)

Subflow

The Activity is refined as a subprocess. This subprocess may be executed synchronously or asynchronously. The subprocess identifiers used are inherited from the surrounding Workflow Model declaration.

- In the case of ***asynchronous execution*** the execution of the Activity is continued after a process instance of the referenced Process Definition is initiated (in this case execution proceeds to any post activity split logic after subprocess initiation. No return parameters are supported from such called processes. Synchronisation with the initiated subprocess, if required, has to be done by other means such as events, not described in this document. This style of subflow is characterised as chained (or forked) sub-process operation.
- In the case of ***synchronous execution*** the execution of the Activity is suspended after a process instance of the referenced Process Definition is initiated. After execution termination of this process instance the Activity is resumed. Return parameters may be used between the called and calling processes on completion of the subflow. This style of subflow is characterised as hierarchic sub-process operation.

Execution Type Keyword	Description
ASYNCHR	Executed asynchronously.
| SYNCHR	Executed synchronously.

Table 3-11: Entity Workflow Process Activity - Implementation as Workflow

Loop

The Activity is refined as a loop (Loop Control Activity) and controls the execution of the Loop repetition (Loop body). The Loop body is connected with the Loop Control Activity by the corresponding Loop connecting Transitions.

A Loop body represents a "bracket" for parts of a Workflow definition that are connected and have only connection to the rest of the definition via the corresponding Loop connecting Transitions.

A LOOP allows expression of repetition ("cycles") in the network in a restricted way. The programming-language like control constructs "WHILE ... DO ..." and "REPEAT ... UNTIL" are supported.

Keyword	Description
WHILE	Restricted to a WHILE Loop
| REPEAT_UNTIL	Restricted to a REPEAT - UNTIL Loop

Table 3-12: Loop Kinds of Entity Workflow Process Activity

The Implementation of a Loop is executed, possibly zero times or repeatedly, until the <loop condition> is evaluated to FALSE (WHILE Loop) or to TRUE (REPEAT_UNTIL Loop), respectively. For the WHILE Loop the test of the condition is performed before (repeatedly) executing the Loop implementation, for the REPEAT_UNTIL Loop afterwards.

3.3.3.5.Performer Relationship

The relationship of the Activity to a (potential) performer is given by the Participant Assignment attribute. It provides a link to the entity Workflow Participant. Default: Any Participant.

The participant assignment may be

1. A participant, which is of one of the permitted participant types of an Organisational Model Definition (ORGANISATIONAL_UNIT, HUMAN, ROLE, SYSTEM) or of the declared Participants.
2. A performer function (a Library Function, built in or extended, delivering a Performer result).

Performer Uniqueness

The question whether the expression evaluation or function invocation results in an empty set of performers or a non unique performer is to be handled by the workflow management system at run

time or, where defined, by the (external) Organisational Model.

In the first case (empty set) the engine may e.g. retry at a later time, or it may signal this to the supervisor of the Process (defined by “Responsible” attribute).

The second case (non-unique) may arise where the performer definition is by function/skill type (defined as “Role”) and/or is an Organisation Unit, which is itself a container for a set of participants. In these situations the approach to participant assignment is local to the WFMS and does not form part of this specification. Common scenarios are:

(i) Where an activity includes multiple work items which may be implemented in parallel (extended attribute “parallel”), separate work items may be presented to a number of performers.

(ii) In other situations the activity may be assigned according to a local load balancing algorithm or presented to multiple potential Performers in their Work lists and assigned to the first accepting participant. (It is the responsibility of the Workflow Engine to provide the appropriate behaviour.

(iii) The assignment of an activity to an OU function (e.g. a department) may result in the activity being offered to all members of the OU and assigned to the first accepting participant or allow the manager of the OU to redirect the activity to a designated departmental member.

In all cases the participant assignments defined within the meta-model and expressed in WPDL only relate Activities to defined Participants (including the use of expressions and defined Functions) and do not differentiate between cases where the defined Participant is atomic (e.g. a person) or not (e.g. a team). The local behaviour of the Workflow Engine and/or the OM Management in managing these situations is not defined.

Scope

The Workflow Participant identifiers used in the PERFORMER attribute have either to be declared in the surrounding Workflow Process definition or are inherited from the surrounding Workflow Model declaration.

3.3.3.6.Simulation Instantiation Attribute

The Instantiation Attribute defines how many times an Activity can be activated for higher throughput (e.g. how many individuals can capture a role). This can be once or many times (multiple).

WPDL Keyword	Value Keywords	Description
INSTANTIATION	ONCE	The Activity can only be instantiated once. Default.
	| MULTIPLE	The Activity can be instantiated multiple times. The possible number has to be specified by an Extended Attribute.

Table 3-13: Entity Workflow Process Activity - Instantiation Attribute

3.3.3.7.Transition Restrictions

Attribute Name	M/O	Restriction Keyword	Data Type	Description
Inline Block information: *	O	INLINE_BLOCK_BEGIN	IDENTIFIER	The Activity is first or last Activity of an Inline Block. *(details see below).* The first Activity of an Inline Block. The identifier is that of the block *(For further details and attributes see below).*
		INLINE_BLOCK_END	IDENTIFIER	The last Activity of an Inline Block.
JOIN description*	O	JOIN	KEYWORD	Specifies that the incoming Transitions of the Activity are JOIN-ed *(details see below).*
SPLIT description*	O	SPLIT	*(see below)*	Specifies that the outgoing Transitions of the Activity are SPLIT *(details see below).*

Table 3-14: Transition Restriction Attributes of Entity Workflow Process Activity

Inline Block

An Inline Block represents a "bracket" for parts of a Workflow definition that are connected and connect to the rest of the definition via the activities having INLINE_BLOCK_BEGIN and the corresponding INLINE_BLOCK_END attributes with the same <block id>. This provides a "light-weight" alternative implementation to the use of a subflow (in which the block activities would be declared as a separate process definition). An inline block identifier is referenced in the same way as an activity identier and may have similar join (prior) and split (post) processing.

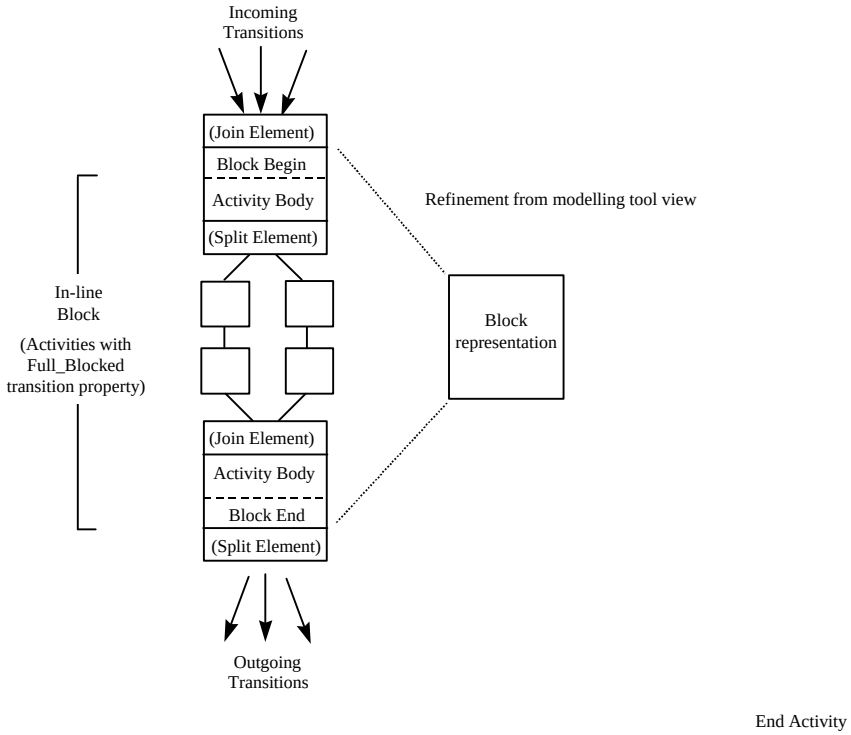

Figure 3-6: ***Inline Block Structure***

Activities within the inline block must follow the restrictions on transition usage as specified below (essentially not to include transitions to activities outside the block). An inline block operates with the same entity instances as the surrounding process definition and therefore does not include any workflow data copy/restore semantics or subflow audit data as would be associated with sub-process invocation.

The keyword `INLINE_BLOCK_BEGIN` is paired by `END_INLINE_BLOCK_BEGIN` at the end of the definition; and inside this body in addition to the Attributes below Extended Attributes are permitted.

Attribute Name	**Restriction Keyword**	**Data Type**	**Description**
Block Name	NAME	`STRING`	Text Used to identify the block.
Block Description	DESCRIPTION	`STRING`	Short textual description of the block.
Icon	ICON	`REFERENCE`	Address (path- and filename) for an icon to represent the block. Allows graphical shrinking of the block body to an Icon.

Table 3-15: Inline Block Attributes of Entity Workflow Process Activity

Scope

The Inline Block Identifier is in the same name space as the Activity Identifier.

JOIN

A JOIN describes the semantics if multiple incoming Transitions for an Activity exist.

JOIN Keyword	Description
AND	Join of (all) concurrent threads within the process instance with incoming transitions to the activity: Synchronisation is required. The number of threads to be synchronised might be dependent on the result of the conditions of previous SPLIT(s).
| XOR	Join for alternative threads: No synchronisation is required..

Table 3-16: *JOIN alternatives of Entity Activity*

The AND JOIN can be seen as a " rendezvous precondition" of the Activity; the activity is not initiated until the transition conditions on all incoming routes evaluate true.

The XOR JOIN initiates the Activity when the transition conditions of any (one) of the incoming transitions evaluates true.

SPLIT

A SPLIT describes the semantics where multiple outgoing Transitions for an Activity exist.

SPLIT Keyword	Data Type	Description
AND		Defines a number of possible concurrent threads represented by the outgoing Transitions of this Activity. If the Transitions have conditions the actual number of executed parallel threads is dependent on the conditions associated with each transition which are evaluated concurrently. *(For Transitions having CONDITION OTHERWISE and other details see below.)*
| XOR	List of IDENTIFIER	List of Identifiers of outgoing Transitions of this Activity, representing. alternatively executed transitions. The decision as to which single transition route is selected is dependent on the conditions of each individual transition as they are evaluated in the sequence specified in the list. If an unconditional Transition is evaluated or transition with condition OTHERWISE this ends the list evaluation.

Table 3-17: SPLIT alternatives of Entity Activity

An AND SPLIT with transitions having conditions may be referred to as "conditional AND", "multiple-choice OR", or "nonexclusive OR", respectively. The number of actual concurrent threads is determined at execution time when evaluating the conditions. Following such an AND SPLIT the process instance (or thread of the process instance) is forked into a number of separate execution

threads which result from the transitions condition evaluation. (Note that no list of identifiers is required since all outgoing transitions from the activity are evaluated and no sequence is necessary.)

If within the AND_SPLIT there is a transition having condition OTHERWISE, then a two-step evaluation is performed. In the first step evaluation is made of all the Transitions except that within the OTHERWISE condition. If none of them (including those having no condition) evaluate to TRUE, then in the second step the same procedure is performed for the Transition with OTHERWISE (only one transition with an OTHERWISE clause is permitted in the list of outgoing transitions from an activity).

An OTHERWISE alternative can be used to guarantee that there is no undefined status from the Process execution (i.e. at least one outgoing transition from an activity will always occur).

3.3.3.8.Conformance Classes

There are Conformance Classes restricting the Activity-Transition Net.

The following Conformance Classes are defined in the Model Definition (see chapter 3.5.3):

- NON-BLOCKED

There is no restriction for this class.

- LOOP-BLOCKED

The Activities and Transitions of a Process Definition (excluding the Transitions connecting a Loop Activity) form an acyclic graph (or set of disjunct acyclic graphs). For cycles only a Loop Implementation of an Activity may be used.

- FULL-BLOCKED

For each JOIN (or respectively SPLIT) there is exactly one corresponding SPLIT (or respectively JOIN) of the same kind, and the Activity of the SPLIT and the corresponding JOIN are also the pairing BEGIN and END activities of an INLINE_BLOCK. In an AND SPLIT no conditions are permitted; in an XOR SPLIT an unconditional or OTHERWISE Transition is required if there is a Transition with a condition (i.e. an undefined result of transition evaluation is not permitted).

3.3.4. Transition Information

Informal Description

The Transition Information describes possible transitions between activities and the conditions which enable or disable them (the transitions) during workflow execution.

A process definition is seen as a network of edges between the Activity nodes (i.e. as a workflow process diagram). All edges are directed and given by a pair of Activities:

(From <node>, To <node>).

The edges of the Activity net may be labelled by ***Transition conditions***. A Transition condition for a specific edge enables that transition if the condition evaluates to TRUE. If no routing condition is specified the Transition behaves as if a condition with value TRUE is present.

If there are multiple incoming or outgoing ("regular", see below) Transitions of an Activity , then

further options to express control flow restrictions and condition evaluation semantics are provided in the Activity entity definition (AND/XOR variants of SPLIT/JOIN).

Transition kind

Two Transition kinds are distinguished, "regular" and Loop-connecting Transitions.

For "regular" Transitions (without using the keyword LOOP) it is possible to define or synchronise multiple (concurrent or alternative) control threads (SPLIT, JOIN) and sequences of Transitions between Activities (cascading Transitions/conditions) and Blocking restrictions.

Loop-connecting Transitions (using the keyword LOOP) allow the expression of cycles in the Transition network. They connect the body of a Loop with the Loop Activity which is implemented by this body (see chapter 3.3.3.4). For all Transitions a FROM part and a TO part are mandatory. Loop conditions are expressed in the loop Activity, not as Transition conditions.

Scope

For the identifiers and names defined in the Transition information the scope is the surrounding Workflow Process Definition (chapter 3.3.2).

The Workflow Activity identifiers used have to be declared in the surrounding Workflow Process Definition.

3.3.4.1.Attributes

Attribute Name	M/O	Keyword	Data types	Description
Identifier	M	TRANSITION	IDENTIFIER	Used to identify the Transition.
Name	O	NAME	STRING	Text used to identify the Transition.
Description	O	DESCRIPTION	STRING	Short textual description of the Transition.
Transition kind	M	*	*	Determines the kind of a Transition. *(see below)*

Table 3-18: Attributes of Entity Transition

Transition Kind

Transition Kind Attribute	M/O	Keyword	Data types	Description
Regular	M	FROM	IDENTIFIER	A "regular" Transition Determines the FROM source of a Transition. (Activity Identifier)
	M	TO	IDENTIFIER	Determines the TO target of a Transition (Activity Identifier)
	O	CONDITION	BOOLEAN *(expression)*	A Transition condition expression based on workflow relevant data. (E.g. *'Contract' = 'SMALL'* OR *'Contract' <$20,000*). Default: TRUE
| Loop Begin Connection	*M*	*FROM LOOP*	*IDENTIFIER*	*A Transition that connects the begin of a Loop body implementing a Loop Activity.* *Determines the FROM source of a Loop Connection Begin Transition. (Activity Identifier)*
	M	*TO*	*IDENTIFIER*	*Determines the TO target of a Loop Connection Begin Transition (Activity Identifier)*
| Loop End Connection	*M*	*FROM*	*IDENTIFIER*	*A Transition that connects the end of a Loop body implementing a Loop Activity.* *Determines the FROM source of a Loop Connection End Transition. (Activity Identifier)*
	M	*TO LOOP*	*IDENTIFIER*	*Determines the TO target of a Loop Connection End Transition (Activity Identifier)*

Table 3-19: Kinds of Entity Transition

3.3.5. Workflow Application Declaration

Informal Description

Workflow application declaration is a list of all applications or tools required and invoked by the workflow processes defined within the process definition or surrounding model. Generic tools may be defined (or, in fact, just named). This means, that the real definition of the tools is not necessary and may be handled by an object manager. The reason for this approach is the handling of multi-platform environments, where a different program (or function) has to be invoked for each platform. WPDL abstracts from the concrete implementation or environment (thus these aspects are not of interest at process definition time).

3.3.5.1.Attributes

This table contains details of the applications which will be invoked at run time. It includes the WAPI (Interface 2/3) conventions for parameters passed to and from third party applications.

Attribute Name	**M/ O**	**Keyword**	**Data type**	**Description**
Identifier	M	APPLICATION	IDENTIFIER	Used to identify the workflow application definition
Name	O	NAME	STRING	Text used to identify an application (may be interpreted as a generic name of the tool).
Description	O	DESCRIPTION	STRING	Short textual description of the application.
Tool Name	O	TOOLNAME	STRING	Name of invoked application
Application invocation Parameters*	O	*(see chapter 3.2.2.2)*		Parameters that are interchanged with the application via the invocation interface.

Table 3-20: Attributes of Entity Workflow Application

3.3.5.2.Invocation Parameters

A Workflow Application declaration may have parameter definitions for the (invocation) parameters as described in chapter 3.2.2.2. and also used within other entities.

The **parameter passing semantics** for invocation is that described in chapter 3.2.2.2.

Concurrency semantics

Copying the invocation IN_PARAMETERS is treated as one atomic operation. The same holds for restoring the invocation OUT_PARAMETERS. Between these two operations no assumption is made about concurrency behaviour.

3.3.6. Workflow Relevant Data

Informal Description

Workflow relevant data represent the variables of a workflow process or workflow model definition. They are typically used to maintain decision data (used in conditions) or reference data values (parameters) which are passed between activities or sub-processes (This may be differentiated from workflow application data, which is data managed or accessed wholly by the invoked applications and which is not accessible to the workflow management system.) The workflow relevant data list defines all data objects which are required by the workflow process. The

attribute TYPE explicitly specifies all information needed for a workflow management system to define an appropriate data object for storing data which is to be handled by an active instance of the workflow process.

Workflow Relevant Data can be defined in a Workflow Process (the Workflow Process Relevant Data) and in a Workflow Model (the Workflow Model Relevant Data). The scopes differ in that the former may only be accessed by entities defined inside that process, while the latter may be used also e.g. to define the parameters of a Process entity.

Where parameters are passed to a called subprocess outside the current model definition (e.g. to support remote process invocation) it is the responsibility of the process designer(s) to ensure that data type compatibility exists across the parameter set.

3.3.6.1.Attributes

Attribute Name	M/O	Keyword	Data type	Description
Identifier	M	DATA	IDENTIFIER	Used to identify the workflow relevant data.
Type*	M	TYPE	*	Datatype.
Name	O	NAME	STRING	Text used to identify the workflow relevant data
Length	O	LENGTH	INTEGER	The length of the data
Description	O	DESCRIPTION	STRING	Short textual description of the data defined.
Value*	O	DEFAULT_VALUE	*	Preassignment of data for run time, *an initial or a function access of appropriate type*

Table 3-21: Attributes of Entity Workflow Relevant Data

Type and Value Attribute

For a Type Attribute value simple and complex data types are permitted. The corresponding DEFAULT_VALUE values have to match this type.

3.3.7. Workflow Participant Specification(Organisational Model)

Overview

This is the static definition of the Workflow-relevant part of an Organisational Model. The interface to the Organisational Model is used in the Activity Definition (describing the performer of an activity) and in the Process Definition (describing the responsible of a process). The meta-model (and WPDL) defines a simple in-built (Minimal) Organisational Model or permits access to an externally defined OM. (via extended library procedures) to reference more complex OM entity relationships.

The Workflow Participant is defined by a type and related information, which is a set of type specific attributes. This definition contains a basic set of 4 Workflow Participant types: an organisational unit, a human, a role, and system (non-interactive execution). A role is used in the sense of abstract actors. During run time these abstract definitions are evaluated and assigned to concrete human(s) and/or program(s).

- The minimal OM declaration does not support relationships between different participant entities and is essentially a list of the identifiers of each of the four participant types.
- An external OM may contain substantial additional information; some functions may be added as required by extended library declarations. As the OM is part of the environment of the Workflow Definition these OM functions are environment functions. This document assumes that they will be defined in the OM referenced in the Process Model surrounding a Process Definition and inherited from there.

In addition to this structural definition, access to the System Environment may be provided via Library Functions that may be used within a Workflow Process Definition (for example current actor or current responsible). Further specific Extended Library Functions may be defined by a vendor or OM definer.

3.3.7.1.Attributes

Attributes (*Declaration*)

For the Minimal Organisational Model Declaration the Participant Type Description attribute (see Table 3-22*) differs from the OM definition in that the Type of a workflow participant is optional, and the Participant type related information is not present.*

The attributes of a Workflow Participant characterise the Participant Type and permit specification of simulation-relevant data.

Attribute Name	M/O	WPDL Keyword	Data Type	Description
Participant Identifier	M	PARTICIPANT	IDENTIFIER	Used to identify the workflow participant definition.
Participant Name	O	NAME	STRING	Text used to identify a performer
Participant Description	O	DESCRIPTION	STRING	Short textual description of a workflow participant.
Participant Type Description *	M	TYPE	*Keyword*	Definition of the type of workflow participant entity. Type of a workflow participant.

Table 3-22: Attributes of Entity Workflow Participant Declaration

Scope

The scope of the identifier of a Workflow Participant entity declaration in a minimal OM Model Declaration is the surrounding entity (Workflow Process Definition or Process Model Definition) in which it is defined.

3.3.7.2.Participant Entity Types

The Participant entity type attribute characterises the participant to be an individual, an organisational unit or an system resource such as a machine.

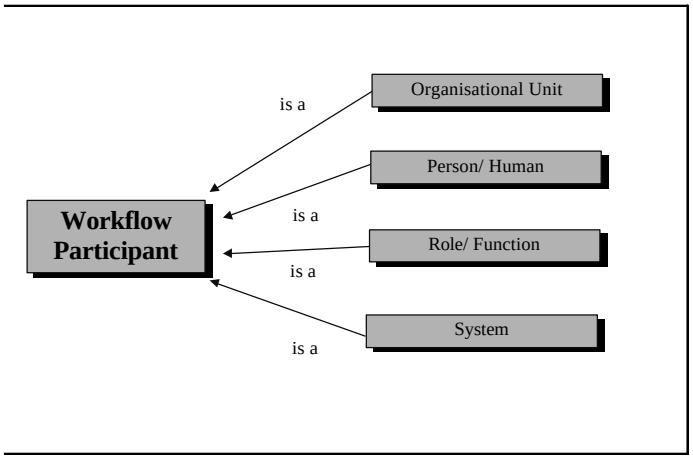

Figure 3-7: Types of Workflow Participant Assignment

WPDL Keyword	Description
ORGANISATIONAL_UNIT	The manager (representing the organisational unit) or all members of an organisational unit get the work item if an organisational unit is addressed.
HUMAN	A specific human participant. (In most workflow processes a human is addressed indirectly by his role, or by his organisational unit rather than directly as a participant.)
ROLE	This type allows performer addressing by a role or skill set. A role in this context is a function a human has within an organisation. As a function isn't necessarily unique, a coordinator may be defined (for administrative purposes or in case of exception handling) and a list of humans the role is related to.
SYSTEM	This type allows addressing a performer as being the system(e.g. as an agent or a machine like an automatic scanner).

Table 3-24: Types of Workflow Participants

3.4. System Environment Access

In the definition of a Workflow Process it is sometime necessary to reference information in the System Environment. For this purpose predefined Library Functions and Procedures are supported. These predefined Library elements provide means to access environment data like Date and Time and system data of the Workflow Management System such as Audit and Process History Data. In the case of Library Functions without parameters specified in this chapter it is also possible to think of them as predefined read-only environment Workflow Relevant Data entities that are permanently updated by the environment (workflow management software or underlying system software).

3.4.1. Built in Library Functions and Procedures

Library functions and procedures provide a means of defining access to a small set of standard dynamic data available (or potentially available) within the surrounding operational environment. Examples of such use are to access:

- system data such as current date and time
- workflow process data such as performers of current activities within a process instance or current duration of process or activity (existence times)
- workflow historical data such as performer of previous activity
- external Organisation Model information such as relationships or organisational hierarchy

The following standard library functions and procedures are defined as (optional) components within the meta-model

3.4.1.1.Built in OM Library Functions and Procedures

This documents assumes that they have to be defined in the OM referenced in the Process Model surrounding a Process Definition and inherited from there.

3.4.1.2.Predefined Date Functions

The following access to environmental Date is provided

function id	return type	parameters	description
current_date	date		Current date is the date at time of execution in conformance with ISO 8601

Table 3-25: Built in Date Library Functions

3.4.1.3.Predefined Process History and Audit Data Functions

The Process History is not part of the Organisational Model. Therefore the reference to previous Performers in the history of a process is handled by the Workflow engine (e.g. by evaluating the Audit Data). Library Functions may be defined to provide access to such historical performer information. (Note, however, that some vendors may implement these functions within an OM which lies outside the current specifications within this version of the document.).

The following access to History and Audit data is provided:

Functions

function	result type	parameters	description
Current_actor	participant		The performer of the current Activity (in case of a human that one who has picked the Activity from his/her worklist (also in the case of automatic execution).
Current_responsible	participant		*The current process Responsible*
Performer_of_last	participant		*The performer of the previous Activity (that would be gained if during execution of that Activity Current_actor was used) in case the Activity has no AND JOIN part, otherwise UNKNOWN*
Current_performer	participant		*The performer as defined in the definition if not an expression, otherwise UNKNOWN*

Table 3-26: Built in History and Audit Library Functions

3.5. Workflow Model

3.5.1. Meta-Model

Multiple process definitions are bound together in a model definition. The Workflow Model acts as a container for grouping together a number of individual process definitions and associated entity data which is applicable to all the contained process definitions (and hence requires definition only once). The Workflow Model meta-model contains the following entity types:

- **Workflow Process Definition**
- **Workflow Participant Specification (OM Specification)**

- **Workflow Application Declaration**
- **Workflow Relevant Data**

as described below.

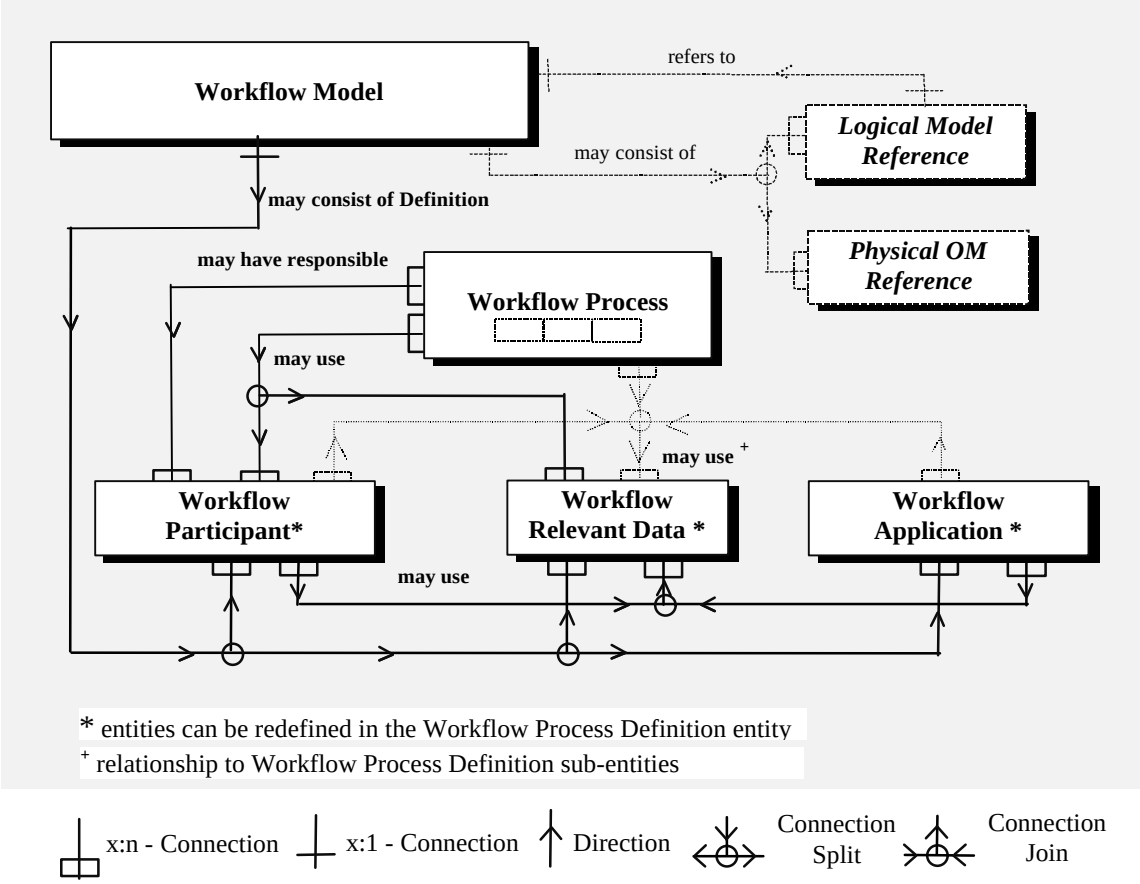

Figure 3-8: Workflow Model Definition Meta Model

3.5.1.1.Informal Description

The meta-model for the Workflow Model identifies the entities and attributes for the exchange, or storage, of process models. It defines various rules of inheritance to associate an individual process definition with entity definitions for participant specification, application declaration and workflow relevant data, which may be defined at the workflow model level rather than at the level of individual process definitions.

The Workflow Participant Specification within the workflow model is either:

a declaration (using the built-in minimal Organisational Model), or is

a full definition using referenced information in an external Organisation Model. Where an external

OM is referenced it is also declared (as an external reference) at the process model level.

The workflow model definition allows the specification of a number of common process definition attributes, which will then apply to all individual process definitions contained within the workflow model. Such attributes may then be omitted from the individual process definitions. (If they are re-specified at the level of an individual process definition this local attribute value takes precedence over the global value defined at the workflow model level.

3.5.1.2.Process Repository

As noted in section 3.1, the process definition import/export interface is assumed to operate to/from a workflow definition repository of some form associated with the workflow management system. The import/export interface is realised by the transfer of files containing WPDL into or out of such repository or by API call (WAPI) allowing the fetching or setting of individual process definition attribute data. This interface specification allows the import or export of process definition data at the level of individual process definitions and workflow models.

The internal interface between the repository and workflow control functions is specific to individual vendor products and does not form part of this standard. It is assumed that separation is provided (for example by version control) between repository usage as a static repository (for persistent, ongoing storage of process definition data) and any dynamic usage (for managing changes to the process execution of extant process instances). This specification only relates to static repository changes. The API interface may potentially be applied to both classes of usage subject to appropriate agreement on conformance classes.

The local storage structure of the process definition repository is not part of the WfMC standard; it is for individual vendors or users to define. Hence the use of a workflow model is defined only as part of the import/export data structures (although there may be obvious advantages in continuing the concept into repository structure). Where a simple process repository structure is used, operating at a single level of process definition, shared information within an imported workflow model may be replicated into each of the individual process definitions at the import interface (and similarly repacked, if required, for process definition export).

3.5.2. Workflow Model Attributes

There are a number of attributes relating to the model itself and other attributes which are common to the process definitions contained within the model. The column ***R*** describes which of the attributes inherited in the Workflow Process entity from the Workflow Model may be redefined there.

Attribute Name	M/O	R	WPDL Keyword	Data Type	Description
Model Identifier	M		MODEL	IDENTIFIER	Used to identify the workflow model.

Attribute Name	M/O	R	WPDL Keyword	Data Type	Description
Model Name	O		NAME	STRING	Text. Used to identify the workflow model.
Model Description	O		DESCRIPTION	STRING	Short textual description of the Workflow Model.
WPDL Version	M		WPDL_VERSION	STRING	Version of the WPDL
Source Vendor ID	M		VENDOR	STRING	Defines the origin of this model definition and contains vendor's name, vendor's product name and product's release number (for instance: "CSE:WorkFlow:4.0" "IBM:FlowMark:2.0" "Ley:COSA:2.0" "SNI:WorkParty:2.0" etc.)
Creation Date	M	R	CREATED	DATE	Creation date of workflow model definition.
Version	O	R	VERSION	DATE	Version of this workflow model definition.
Author	O	R	AUTHOR	STRING	Name of the author of this workflow model definition. (The one who entered it into the repository)
Character set	O	R	CHARACTERSET	STRING	Name of the character set, e.g. OEM for DOS, ANSI for Windows, ...
Code page	O	R	CODEPAGE	STRING	The codepage used for the text parts
Country key	O	R	COUNTRY_KEY	STRING	nnn as country number
Responsible	O	R	RESPONSIBLE	PARTICIPANT *(expression)*	Workflow participant, who is responsible for this workflow process; the supervisor during run time (a human). Link to entity workflow participant. Workflow participant, who is responsible for this workflow of this Model definition (usually an Organisational Unit or a Human). It is assumed that the responsible is the supervisor during run time. Default: Initiating participant.
Publication Status*	O	R	STATUS	*Keyword*	Status of the Workflow Model Definition. *(see chapter 3.3.2.1)*
Documentation	O		DOCUMENTATION	REFERENCE	Operating System specific path- and filename of help file/description file.
Units					Units used in Simulation Data

Attribute Name	M/O	R	WPDL Keyword	Data Type	Description
	O		PRIORITY_UNIT	STRING	Priority unit: A text string with user defined semantics.
	O		DURATION_UNIT	Keyword	Duration unit.
	O		COST_UNIT	STRING	Cost unit: Usually expressed in terms of a currency.
Conformance Class*	O		*CONFORMANCE_ CLASS*	Keyword	Describes the Conformance Class to which the definitions in this model are restricted.
Extended library*	O		*(see chapter 3.5.5)*		Functions and procedures in an extended library *(for details see chapter 3.5.5)*
External. Model Reference*	O		EXTERNAL_ MODEL_ REFERENCE	(see below)	List of references to external models *(for details see below)*

Table 3-27: Attributes of Entity Workflow Model

Values of DURATION_UNIT:

CALENDAR_DAY	A number that can be added to a Date and delivers another Date.

Table 3-28: Possible Value of Duration Unit

3.5.3. Conformance Class

The following conformance classes are supported: The specified class applies to all the contained process definitions, unless it is re-defined locally at the process definition level.

Class Name	Class Keyword	Description
Block restricted	FULL-BLOCKED	The network structure is restricted to proper nesting of SPLIT/JOIN and LOOP.
Loop restricted	LOOP-BLOCKED	The network structure is restricted to proper nesting of LOOP.
Unrestricted	NON-BLOCKED	There is no restriction on the network structure. This is the default.

Table 3-29: Conformance Classes of Entity Workflow Model

Further details are described in chapter 3.3.3.8.

3.5.4. External Model Reference

External model reference allows reference to definitions contained within other Model Definitions. These may either be an external Workflow Model defined in WPDL or an external Organisational Model (e.g. an external system like a personal management system with appropriate interface).

Attribute Name	M/O	WPDL Keyword	Data Type	Description
Logical Reference	M	WM	IDENTIFIER	A Model Identifier. Logical reference to a Model
	O	*RESTRICT_TO*	List of IDENTIFIER	List of Identifiers of Workflow Relevant Data defined in the referenced Model Definition. Restricts access to Workflow Relevant Data to those listed. Default: No restriction
Physical reference	M	OM	REFERENCE	Physical reference to a Model (e.g. reference to an external OM)

Table 3-30: External Model Reference Attributes of Entity Workflow Model

The keyword EXTERNAL_MODEL_REFERENCE is paired by END_EXTERNAL_MODEL_REFERENCE at the end of the definition; and inside this body Extended Attributes are permitted.

3.5.4.1.Redefinition and Scoping

The possibility of redefining attributes and meta-model entities and referencing external models introduces the principles of scope and hierarchy into the WPDL (and process repository) structures.

(i) Workflow relevant Data

Workflow process relevant data has a scope that is defined by the directly surrounding meta-model entity and is not nested. The visibility of its identifier is also defined by that entity.

(ii) Attributes

Attributes including extended attributes have a scope that is defined by the directly surrounding meta-model entity and are nested, i.e. may be redefined at a lower level. Example: The name attribute is redefined in each entity definition. The visibility of extended attribute identifiers is within the particular entity and all sub-entities unless the identifier is redefined in a sub-entity.

(iii) Workflow participants and applications

Workflow participants and applications have a scope and visibility equivalent to extended attributes.

All referenced workflow relevant data and extended attributes have to be defined in the scope where they are used, at least in the same model. However, for process ids, participant ids, application ids and library element ids a further mechanism is introduced by the external model possibility. This mechanism e.g. allows invocation of subprocesses defined in another model.

(iv) Externally referenced entities

For a referenced external model entity that needs itself reference to entities and their identifiers in models defined in its external model reference clause the mechanism is started with the root in that model. That guarantees that no conflict takes place if the invoking process has an entity with the same id which the definer of the referenced model cannot be aware of.

The described mechanism of external model reference provides high flexibility for workflow designers and administrators. One can separate organisation descriptions (participant entities) and process definitions in separate models, one can add a new release of a process description or add a new process definition sharing the rest of the definition of previously defined and exchanged models without resubmitting the whole context etc.

3.5.5. Extended Library

Informal Description

The Extended Library attribute may be used in the Workflow Process and the Workflow Model entity. It allows declaration of Library Functions and Procedures. It may contain two parts, function and procedure declarations, which have further attributes including Extended Attributes.

3.5.5.1.Attributes

Attribute Name	M/O	WPDL Keyword	Data Type	Description
Library element identifier	M	FUNCTION	IDENTIFIER	Identifier. Used to identify the library element Identifies a library function
		| PROCEDURE	IDENTIFIER	Identifies a library procedure
Result type	M	RESULT	*	A plain data type to denote the result type (for a Library Function only)
Name	O	NAME	STRING	Text. Used to identify the library element.
Description	O	DESCRIPTION	STRING	Short textual description of the library element.
Parameters	O	(see chapter 3.2.2.2)		Parameters which are passed through to the Library Function or Procedure. For a procedure parameters may contain result values.

Table 3-31: Attributes of Library Functions and Procedures

The keywords FUNCTION and PROCEDURE are paired by END_FUNCTION and END_PROCEDURE at the end of the definition; and inside this body Extended Attributes are permitted.

4. Proposed WPDL Grammar

4.1. WPDL at a glance

The workflow process definition language (WPDL) is a language for describing workflows as an ASCII character stream (which may be a flat file or a string) using keywords (like WORKFLOW, ACTIVITY, DESCRIPTION etc.) for specifying objects, attributes and relationships and using variable parts in the grammar (string constants, and placeholders like process relevant data, etc.) for specifying their names and values.

In summary:

- The grammar is given in EBNF (Extended Backus Naur Form)
- Keywords are used to start an entity description (the entities represented in WPDL are contained in the minimum meta model)
- Keywords are written in uppercase letters
- Keyword - value pairs are used to specify attributes
- Keywords are used to specify relations to other entities
- Attributes and relations are optional
- Attributes and relations of entities are identified by keywords and don’t have to appear in order
- Relations between two entities are defined on either side of the participating entities
- Tokens within an expression are separated by one or more whitespace characters
- Keywords are taken from the WfMC glossary
- Comments are supported between "/* and "*/” and after "//” (for the rest of the line)

Characteristics:

The WPDL language offers

- a minimum number of pre-defined entities (see minimum meta model)
- a minimum number of pre-defined relations between entities
- a number of pre-defined attributes (using keywords)
- additional generic attributes (for vendor specific attributes that are not pre-defined)
- additional generic relations (for vendor specific relations that are not pre-defined)
- additional generic data objects (for data objects that are not in the minimum meta model)

Recommendations:

- Only use a generic data object, if it cannot be mapped to a pre-defined entity.
- Only use a generic relation, if it cannot be expressed by a pre-defined relation.
- Only use a generic attribute, if it cannot be found in the table of pre-defined attributes.

4.2. WPDL Grammar and Language Constructs

On the following pages we define the grammar of WPDL. We do this by using a BNF-like format (BNF = Backus, Naur Form). Before introducing the syntax, we explain some general BNF rules which are used throughout this grammar, and introduce some generic and common language constructs.

4.2.1. WPDL Description Method

4.2.1.1.Metalanguage

The components of this grammar (the metalanguage) consist of:

```
·Symbols                    <example_symbol>
·Keywords                   EXAMPLE_KEYWORD
·Production sign            ::=
·Special characters    [ ] | / *
```

The Workflow Process Definition Language is defined as a set of productions.

On the left hand side of the production a symbol appears which is not part of the language. This symbol summarises the components on the right hand side of the production sign. Therefore the right hand side of a production defines a rule for the development of the symbol on the left hand side.

If a symbol appears on the left hand side of a production, it can be substituted for by the contents of the corresponding rule. The right hand side is a combination of symbols, keywords and special characters.

The keywords are the central parts of the language separated by blanks (white spaces). Keywords are case-sensitive, i.e. the usage of upper and lower case letters has to be considered.

A keyword or symbol (or combinations of both) appearing in square brackets ("[" and "]") indicates the construct is optional. The special character "|" implies exclusivity, i.e. one decides between the option before or behind the "|" character.

The special character combinations "/*" and "*/" indicate the part between these combinations and "//" that the subsequent part of the line up to the end are comments.

4.2.1.2.Conformance Relationship

The WPDL is constructed with three conformance classes in mind, which are reflected in the language in different ways.

Mandatory parts to be supported by all vendors are those that are neither of the two subsequent.

Optional parts are those not supported by all vendors, but defined in the WPDL. Hints for conformance optional parts are included in the grammar as comment: ("// optional"). Annex A (chapter 5) contains a more elaborate list.

Also provision is made to allow for vendor specific parts. These are "extended" parts of the language. A detailed description of these parts has to be made by the vendor. A further description is provided in chapter 3.1.5.

4.2.1.3.Restrictions

The WPDL elements are described in the form of Syntax rules (with associated semantics) and restrictions to these rules.

If these restrictions are not fulfilled, then this configuration is outside the scope of what the WPDL defines, and the semantics then is not specified by this standard (i.e. is vendor dependent).

4.2.1.4.Special Symbol Conventions for Tokens

Some kinds of symbols, the tokens, are not further decomposed in the WPDL.

The basic symbols are those whose descriptions are outside the WPDL.

Keywords characterise the WPDL process model description, their parts and attributes (model-relevant tokens).

Some kinds of tokens used for data types and expressions have a WPDL representation containing further keywords and special characters. To allow easier distinction of these tokens from the meta language elements and the model-relevant keywords to provide for easier extensions we have introduced special terminal productions and added a chapter describing the WPDL representation of these special symbols.

By convention these special symbol classes are:

Operator symbols terminate with letters "Op" (e.g. `<NotOp>`)

Constant symbols terminate with an upper case letter "C" (e.g. `<BooleanC>`)

Bracket symbols terminate with an upper case letter "B" (e.g. `<OpenArrayB>`)

Type symbols are written in uppercase letters and terminate with an upper case letter "T" (e.g. `<INTEGER-T>`)

Other terminal symbols are written in uppercase letters (e.g. `<UPTO>`)

There are other special symbols for which further conventions exist:

Symbols that denote lists terminate with "list" or " List" (e-.g. `<Activity List>`, `<roles description list>`), and symbols denoting vendor-defined parts begin with "extended" (e.g. `<extended attribute list>`).

4.2.2. Basics, Data Types, and Expressions

4.2.2.1.Basics

In the grammar some symbols are basic and used to define other symbols. The formal definition of these symbols has been left out here, i.e. they are tokens to the WPDL. Their meaning and representation are defined elsewhere.

Basic Data

This is valid for the following <basic data> symbols:

```
<basic data>          ::=   <string> | <float> | <integer> | <reference> | <date>
```

where

<string>::= Sequence of alphanumeric and special characters, beginning with ' " ' (double quote) and ending with ' " ' (e.g. "example"). Double quotes within a string must be preceded by double quotes (e.g. "(e.g. ""example"")"). Maximum length 1024 characters incl. null bytes and quotes.

<float> ::= Number with decimal point - not more precise than the 64 bit IEEE format.

<integer> ::= (Signed) integer - representable in 32 bit.

<reference> ::= A string that holds references to external objects (e.g. filenames like "c:\test\test.exe", mail addresses, urls, ...:)

<date> ::= A date in the format YYYY-MM-DD [hh:mm[:ss]]; default (as defined by ISO 8601/ EN 28601) is:

YYYY is the year in the usual Gregorian calendar,

MM is the month of the year between 01 (January) and 12 (December), and

DD is the day of the month between 01 and 31,

hh is the number of complete hours that have passed since midnight (00-24),

mm is the number of complete minutes that have passed since the start of the hour (00-59), and

ss is the number of seconds since the start of the minute (00-60).

The value 60 for ss appears only in case of an inserted leap second into an atomic time scale like UTC in order to keep it synchronized with a less constant astronomical time scale like UT1.

The hour value 24 is only possible when the minute and second values are zero.

Identifiers

An identifier is a basic symbol:

<identifier> ::= Sequence of letters, digits, underscores and dots; starts with a letter or underscore, and surrounded by single quotes.

Special identifiers are <data id> which represents workflow process relevant data, and <function id> and <procedure id> which represent a library function and procedure, respectively.

Cardinal

A cardinal is a basic symbol used within complex data structure:

<cardinal> ::= Unsigned (positive) integer including 0 - representable in 32 bit

Blank

A blank is a basic symbol:

<blank> ::= Standard separator between WPDL tokens (usually white space, maybe in combination with cr and/or lf).

4.2.2.2. Plain and Complex Data Types

Plain data

Plain data are basics, boolean and performers:

```
<plain data>          ::=   <basic data>
                            | <boolean> | <performer>
```

A data instance of a boolean type, a <boolean>, is one having one of the values true or false. Although the internal representation of these values is not defined in the WPDL (usually 1 and 0), they match with the constant representations TRUE and FALSE, respectively.

A data instance of a performer type, <performer>, is one having a value of a declared workflow participant.

Definition

```
<plain data type>     ::=   <basic data type>
                            | <boolean type> | <performer type>
```

The types of <basic data>, where correspondence is provided by the matching letter sequence ignoring case and terminating "-T", are:

```
<basic data type>     ::=   <STRING-T> | <FLOAT-T> | <INTEGER-T>
                            | <REFERENCE-T> | <DATE-T>
```

The types of <boolean> and <performer>, respectively, are:

```
<boolean type>                  ::=   <BOOLEAN-T>          // the type of a <boolean>
<performer type>                ::=   <PERFORMER-T>        // the type of a <performer>
```

Constants and Initials

Simple constants are used in expressions.

```
<simple constant>                  ::=   <basic data> | <boolean constant>
      <boolean constant>           ::=   <BooleanC>
```

Initials are used to initiate workflow relevant data and extended attributes.

```
<plain initial>                    ::=   <simple constant>
                                         | <PerformerC>
```

Complex data

Complex data permits definition of arrays, records, enumerations, lists or a superset of them within extended attributes or workflow process relevant data, and provides means to access the data defined.

Definition

```
<complex data type>    ::=   <plain data type>
                           | <RECORD> <Member_list> <END>   // a record type
                           | <ARRAY>
                             <OpenArrayB> <cardinal> <UPTO> <cardinal> <CloseArrayB>
                             <OF> <complex data type>      // an array type
                           | <ENUM> <element list> <END>    // an enumeration type;
                           | <LIST> <OF> <element type>     // a list data type
                                                  // has a dynamic number of elements

      <Member_list>                ::=   <Member> [<Member_list>]
      <Member>                     ::=   <complex data type> <Midlist>
      <Midlist>                    ::=   <Member id> [<Midlist>]
      <Member id>                  ::=   <identifier>                          // of matching type
      <element list>               ::=   <element> [<element list>]      // of the same type
      <element>                    ::=   <complex initial>       // of appropriate type
      <element type>               ::=   <complex data type>
```

Initials

```
<complex initial>                  ::=   <plain initial> // enumeration: a defined element
                                         | <OpenArrayB> <element list> <CloseArrayB>
                                            // number and type of elem. matching the array def.
                                         | <OpenB> [<element list>] <CloseB>
                                            // number and type sequ. of elem. matching the record def.
                                            // or type of elem. matching the list def. (maybe empty)

<initial>                          ::=    <complex initial>
```

4.2.2.3.Expressions

Logical or arithmetic expression define any kind of conditions (loop condition, transition condition) or are used in parameters, respectively.

The syntax of expressions is described in a simplified notation, augmented by a precedence list and an applicability table that describe context and type dependent restrictions.

```
<expression>                    ::=   <RelExpression> [<BooleanOp> <expression>]
<BooleanOp>                     ::=   <ANDOp> | <OROp>
<RelExpression>                 ::=   <ArExpression> [<RelationalOp> <RelExpression>]
<ArExpression>                  ::=   <Unary> [<ArithmeticOp> <ArExpression>]
<Unary>                         ::=   [<NotOp>] <Primary>
                                      | <UMinusOp> <Primary>
<Primary>                       ::=   <VarReference>
                                      | <OpenB> <expression> <CloseB>
                                      | <PrimaryConstant>
                                      | <function access>
                                      | PARTICIPANT <participant id>
<PrimaryConstant>               ::=   <simple constant>
<VarReference>                  ::=   <data id>              // of appropriate type
                                       [<VarQualifier list>]
                                                             // for record and array access
      <VarQualifier list>       ::=   <VarQualifier> [<VarQualifier list>]
      <VarQualifier>            ::=   <OpenArrayB> <ArExpression> <CloseArrayB>
                                                             // array access
                                      | <PERIOD> <identifier>
                                                             // record access
<function access>               ::=   <function id> <OpenB> <CloseB>
                                      | <function id> <parameter map list>
                                                  // function built-in or extended
<ActualParameterlist>           ::=   <ActualParameters> [<COMMA> <ActualParameterlist>]
      <ActualParameters>        ::=   <expression>

<condition>                     ::=   <expression>           // delivers a boolean
<integer expression>            ::=   <expression>           // delivers an integer
<string expression>             ::=   <expression>           // delivers a string
<performer expression>          ::=   <expression>           // delivers a performer
```

Expressions are composed as a sequence of operators and operands. All operators are left associative. The defined **precedence order** is (in order of increasing tightness, using the Token Representations defined in the subsequent chapter):

operator
OR
AND
= != < > <= >=
- +
* /
unary: NOT ! -

Table 4-1: Operator Precedence

Parentheses also control evaluation order.

Applicability table for operators and operands of types, where arithmetic are integer and float data types:

operator	**result type**	**first operand type**	**second operand type**

AND OR NOT	boolean	boolean	boolean (except NOT)
= != < > <= >=	boolean	string	string
	boolean	arithmetic	arithmetic
	boolean	date	date
= !=	boolean	reference	reference
	boolean	performer	performer
	boolean	binary	binary
- + * /	arithmetic	arithmetic	arithmetic (except unary -)
+	date	integer	date
-	integer	date	date
+	string	string	string

Table 4-2: Operator Applicability

Plus applied to string operands is string concatenation. The integer in date arithmetic is a duration of an appropriate unit.

Built-in coercions

For arithmetic operators combining float and integer data types the standard conversions hold.

For other type conversions either extended coercions have to be defined (by a mechanism not described here) or special library functions have to be declared.

4.2.2.4. Token Representations in WPDL

Some tokens (terminal symbols) map to a representation defined in the WPDL (similar to keywords). The following productions describe the representation of those tokens that have a WPDL representation and are not workflow relevant keywords.

Operator symbols

```
<ANDOp>                 ::=   AND
<OROp>                  ::=   OR
<NotOp>                 ::=   NOT | !                 // alternative representations
                                                      // with the same semantics
<RelationalOp>          ::=   =  |  !=  |  <  |  <=  |  >  |  >=
<ArithmeticOp>          ::=   +  |  -  |  *  |  /
<UMinusOp>              ::=   -
```

Constant symbols

```
<BooleanC>              ::=   TRUE | FALSE
<PerformerC>            ::=   UNKNOWN
```

Bracket symbols

```
<OpenB>                 ::=   (
<CloseB>                ::=   )
<OpenArrayB>            ::=   [             // to be distinguished from metasymbol [
<CloseArrayB>           ::=   ]             // to be distinguished from metasymbol ]
```

Basic type symbols

```
<STRING-T>              ::=   STRING
<FLOAT-T>               ::=   FLOAT
<INTEGER-T>             ::=   INTEGER
<BOOLEAN-T>             ::=   BOOLEAN
<REFERENCE-T>           ::=   REFERENCE
<DATE-T>                ::=   DATE
<PERFORMER-T>           ::=   PERFORMER
```

Other symbols

```
<END>                   ::=   END
<ARRAY>                 ::=   ARRAY
<RECORD>                ::=   RECORD
<ENUM>                  ::=   ENUM
<LIST>                  ::=   LIST
<UPTO>                  ::=   ...
<OF>                    ::=   OF
<PERIOD>                ::=   .                     // the record selector
<COMMA>                 ::=   ,
<COLON>                 ::=   :
```

4.2.3. Common Constructs

4.2.3.1. Overall WPDL Appearance

The WPDL description of a workflow model, which is a WPDL entity itself, is a sequence of meta model entity descriptions. The individual entities are characterised by keywords. For convenience of document structuring some parts of the entity descriptions are grouped into separate WPDL clauses (e.g. model definition header).

The description of each entity is a sequence of an entity keyword, followed by an identifier, an attribute list, and an end-entity keyword. The model entity in addition includes a list of meta-model entity descriptions before the end-entity keyword.

An attribute is defined as:

```
<attribute>          ::=   <attribute keyword> <attribute description>
```

An attribute is either a predefined attribute or an extended attribute in an <extended attribute list>.

4.2.3.2. Extended Attributes

In addition to the predefined attribute list there exists a generic symbol <extended attribute list>. This allows every vendor of a workflow process definition tool to specify their own set of attributes for the main objects of the meta-model (assuming it is not covered by those already defined). We believe that this is a suitable procedure and will allow most vendors to make use of the WPDL within a very short time-scale.

```
<extended attribute list>          ::=   EXTENDED_ATTRIBUTE
                                               <attribute id>
                                               <attribute type>
                                               <attribute value>
                                               [<description>]
                                         [<extended attribute list>]

      <attribute id>               ::=   <identifier>
      <attribute type>             ::=   <complex data type>
      <attribute value>            ::=   <initial>                        // of matching type
                                         | <function access>              // of matching result type
      <description>                ::=   <string>
```

4.2.3.3.Parameters

A parameter in the context of the WPDL is defined by workflow process relevant data; a parameter list is the aggregation of parameters in a list. Parameters are used to be passed along between process and subprocess, between (sub)process and application etc..

Generic formal parameter definition

Formal parameters are part of the attribute sequence in the WPDL definition of Workflow Process Definition and Workflow Application: We distinguish call input parameters and call output parameters.

The generic form of the part in the attribute sequence is:

```
<formal parameters>                ::=
                                   [IN_PARAMETERS         <parameter list>]
                                                          //call input parameters  (1)
                                   [OUT_PARAMETERS        <parameter list>]
                                                          //call output parameters (2)
      <parameter list>             ::=   <parameter> [<parameter list>]
      <parameter>                  ::=   <data id>       // workflow  relevant data
```

Formal-actual parameter mapping

The mapping of actual to formal parameters during invocation of metamodel entities is defined by a parameter map list:

```
      <parameter map list>         ::=   <OpenB> <parameter map> <CloseB>
      <parameter map>              ::=   <ActualParameterlist>
```

The `<ActualParameterlist>` maps the actual to the formal parameter in sequence, i.e. the first actual maps to the first formal, the second actual maps to the second formal etc. The semantics is defined for formal and actual parameter lists holding the same number of parameters. Alternatively an extended parameter mapping semantics may be specified vendor specific (e.g. setting to zero for missing parameters, ignoring surplus operators etc.). The mechanism for that extension is not provided here.

In case the actual parameter is an expression, the expression is evaluated and buffered by the Workflow engine, and the contents of this buffer is used for formal-actual mapping. How the buffering and mapping is performed is outside the scope if this document.

4.3. WPDL

In this chapter we describe the workflow model that is built up of meta model entity descriptions and attributes. Parts of the description are mandatory, while others are optional (included in square brackets). However, it has to be mentioned that omitting the optional parts completely does not provide a useful model.

4.3.1. Workflow Model

The key rule of the Workflow Process Definition Language is set out below. Each symbol stated on the right-hand side can be considered as a separate section within the resulting workflow model.

It is possible to define several processes within one workflow model which may share the same tools and participants. We recommend creating one workflow model per business process which should contain all the necessary workflow processes as well as all the associated tools and workflow participants, although it is not required. Also it is possible to define just parts of one process definition or common parts of several processes within one workflow model (e.g. a workflow participant list or a workflow application list).

The header of the workflow model definition has to occur once at the very beginning of the model and may be repeated in each process definition separately as workflow process definition header.

```
<Workflow Model>        ::=   MODEL <model id>
                              <Workflow Model Definition Header>
                              [<conformance class declaration>]
                              [<extended library declaration>]
                              [<external model declaration>]
                              [<Workflow Participant Specification>]
                              [<Workflow Application List>]
                              [<Workflow Relevant Data List>]
                                                    // To allow parameter definition for
                                                    // Applications, Library elements etc.
                              [<Workflow Process Definition>]
                              END_MODEL

                        <model id>                  ::=    <identifier>
```

4.3.1.1. Workflow Model Definition Header

The workflow model definition header keeps all information central to a workflow model such as wpdl version, source vendor id, etc.

```
<Workflow Model Definition Header> ::=
                              WPDL_VERSION                <wpdl version>
                              VENDOR                      <source vendor id>
                              CREATED                     <creation date>
                              [NAME                       <name>]
                              [DESCRIPTION                <description>]
                              [<redefinable header>]
                              [DOCUMENTATION              <documentation>]
                              [PRIORITY_UNIT              <unit>]
                              [DURATION_UNIT              CALENDAR_DAY]
                                                          // Calendar_day: A number that can be added to a Date
                                                          and delivers another Date.

                              [COST_UNIT                  <unit>]
                              [<extended attribute list>]

      <name>                        ::=   <string>
      <wpdl version>                ::=   <string>
      <source vendor id>            ::=   <string>
      <creation date>               ::=   <date>
      <version>                     ::=   <string>
      <unit>                        ::=   <string>              // valid for all units
```

The attributes defined here refer to the attributes of the entities within the WPDL meta-model. These attributes should not be considered as complete; in fact, it mainly contains the attributes which are required by the members of WG 1. Nevertheless we think the list should be sufficient for most vendors of workflow process definition tools.

4.3.1.2. Redefinable Header

The `<redefinable header>` covers those header attributes that may be defined in the workflow definition header and may be redefined in the header of any process definition. In case of redefinition scoping rules hold.

```
<redefinable header>   ::=
                              [AUTHOR                     <author>]
                              [VERSION                    <version>]
                              [CHARACTERSET               <characterset>]
                              [CODEPAGE                   <codepage>]
                              [COUNTRY_KEY                <country key>]
                              [RESPONSIBLE                <responsible>]
                              [STATUS                     <publication status>]

      <author>                      ::=   <string>
      <characterset>                ::=   <string>
      <codepage>                    ::=   <string>
      <country key>                 ::=   <string>
      <responsible>                 ::=   <participant assignment>
                                            // usually an organisational unit or a human
```

```
<publication status>          ::=   UNDER_REVISION | RELEASED | UNDER_TEST
```

CREATED is also a redefinable attribute. However, it is mandatory for the workflow model definition header.

4.3.1.3. Conformance Class Declaration

The <conformance class declaration> allows description of the conformance class to which the definitions in this model definition are restricted.

```
<conformance class declaration>
                                  ::=   CONFORMANCE_CLASS <conformance class list>
<conformance class list>          ::=   <graph class conformance list>
                                                  // for future extensions
      <graph class conformance list>
                                  ::=   FULL-BLOCKED            // Only proper nesting
                                        | LOOP-BLOCKED   // Proper Nesting for Loops
                                        | NON-BLOCKED           // No proper nesting required
```

4.3.1.4. Library Functions and Procedures

Library functions and procedures are those that are immediately bound to the workflow engine and are executed without using interface 2. Functions are used in expressions, procedures are invoked in workflow process activities.

Library elements (functions and procedures) are either predefined (built-in, see chapters 3.4.1) or part of an extended library declaration:

```
<extended library declaration> ::=
                                  LIBRARY
                                        <library element list>
                                  END_LIBRARY

      <library element list>      ::=   <library function> [<library element list>]
                                        | <library procedure> [<library element list>]
      <library procedure>         ::=
                                  PROCEDURE             <procedure id>
                                        [NAME           <name>]
                                        [DESCRIPTION    <description>]
                                        [<formal parameters>]   // interface of procedure
                                        [<extended attribute list>]
                                  END_PROCEDURE
      <library function>          ::=
                                  FUNCTION              <function id>
                                        RESULT          <result type>
                                        [NAME           <name>]
                                        [DESCRIPTION    <description>]
                                        [<formal parameters>]   // interface of function
                                        [<extended attribute list>]
                                  END_FUNCTION
      <procedure id>              ::=   <identifier>
      <function id>               ::=   <identifier>
```

```
<result type>                ::=   <plain data type>
```

4.3.1.5. External Model Reference

External model reference allows referencing definitions in another Workflow Model Definition or in other systems providing an Interface to the Workflow Management system (e.g. a legacy Organisation Description Management Tool).

```
<external model declaration> ::=
                                    EXTERNAL_MODEL_REFERENCE
                                          <external model reference>
                                          [<extended attribute list>]

                                    END_EXTERNAL_MODEL_REFERENCE
                                    [<external model declaration>]
      <external model reference>
                                    ::=   <logical model reference>
                                          | <physical model reference>
      <logical model reference>
                                    ::=   WM <model id>
                                           [<Access restriction part>]     // optional
      <physical model reference> ::=OM <reference>
                                                        // e.g. reference to an external OM
```

// Model reference filter

<Access restriction part> *::=* *RESTRICT_TO <parameter list>*

Restriction:

*(**Restricted Data Access:**) If a Model has an <Access restriction part>, then only those Workflow Relevant Data of the <logical model reference> which are listed in its <parameter list> may be used in this <Workflow Process Definition>.*

4.3.2. Workflow Process Definition

The <Workflow Process Definition> defines the elements that make up a workflow.

```
<Workflow Process Definition> ::=
                        WORKFLOW <process id>
                              <Workflow Process Definition Header>
                              [<extended library declaration>]
                              [<formal parameters>]                          // for use as subprocess
                              [<extended attribute list>]
                              [<Access restriction part>]          // optional
                              <Activity List>
                              <Transition Information List>
                              [<Workflow Participant Specification>]
                              [<Workflow Application List>]
                              [<Workflow Relevant Data List>]
                        END_WORKFLOW
                              [<Workflow Process Definition>]

      <process id>                  ::=   <identifier>
```

Explanation:

The Begin (End) Activities of a Process Definition are those that are not contained in the TO (FROM) part of a Transition of this Process Definition and are not referenced in a FROM LOOP (TO LOOP).

If there are multiple Begin Activities of a Process, then they are started concurrently when a Process instance execution starts.

// Entity reference filter and Parameters

Restriction:

*(**Restricted Data Access:**) If a Process has an <Access restriction part>, then only those global Workflow Relevant Data (defined in the <Workflow Model> Definition or listed in the <external model declaration>) which are listed in its <parameter list> may be used in this <Workflow Process Definition>.*

*(**Parameter Restriction:**) If there is an <Access Restriction part> then the Workflow Relevant Data referenced in the <formal parameters> have to be included in the <Access Restriction part>.*

Explanation:

Access Restrictions provide an interface to Data that extends the scope of a Process. If a process has parameters, then the Workflow Relevant Data referenced in the formal parameter definition have to be also included in the Access Restriction part. The semantics is as follows:

(1) If a Workflow Relevant Data identifier is included in the Parameter list, then a (local) instance is created when the Process is instantiated. Formal-actual parameter mapping is performed to this instance.
(1a) If the process is invoked as a subprocess, then the actual parameters are those provided by the invoking Activity.
(1b) If the process is directly invoked, i.e. not as a subprocess, then the Workflow Engine is responsible for handling the actual parameters. These Workflow Relevant Data instances then would be manipulated (written before, read after process execution) by appropriate means.

(2) If a Workflow Relevant Data identifier referenced inside a Process Definition is neither defined in the <Workflow Relevant Data List> of this process nor in its parameter list then the following semantics is assumed:
(2a) If the process is not invoked as a subflow then the *instance of the Workflow Relevant Data is created when the Process is instantiated.*

(2b) If it is invoked as a subprocess then it has to be checked if the work flow relevant data instance already exists (i.e. it has been created in a parent process instance or in a parent of the parent …). If this is not the case then it is created when this Process is instantiated.

4.3.2.1. Workflow Process Definition Header

The workflow process definition header keeps all information specific for a process definition such as process version, priority, duration of validity, etc.

```
<Workflow Process Definition Header> ::=
                            [CREATED                    <creation date>]
                            [NAME                       <name>]
                            [DESCRIPTION                <description>]
                            [<redefinable header>]
                            [PRIORITY                   <priority>]
                            [LIMIT                      <duration>]
                            [VALID_FROM                 <date>]
                            [VALID_TO                   <date>]
                            [CLASSIFICATION             <classification>]
                            [<time estimation>]
                            [DOCUMENTATION              <documentation>]
                            [ICON                       <icon identifier>]


      <process name>              ::=   <string>
      <priority>                  ::=   <integer>
      <classification>            ::=   <string>
      <documentation>             ::=   <string>
      <icon identifier>           ::=   <string>
      <time estimation>           ::=   [WAITING_TIME    <duration>]
                                        [WORKING_TIME <duration>]
                                        [DURATION       <duration>]
      <duration>                  ::=   <integer>
```

Restrictions:

*(**LIMIT**:) If the value CURRENT_DATE plus LIMIT is reached, then this configuration is outside the scope of what the WPDL defines, and the semantics then is not specified by this standard (i.e. is vendor dependent). It is assumed that in this case at least the Responsible of the current process is notified of this situation.*

4.3.3. Workflow Process Activity

The following rule will be used to describe all necessary activities. In addition it allows expression of further condition evaluation and structure restrictions of Transitions.

```
<Activity List>    ::=
                     ACTIVITY                      <activity id>
                          [NAME                    <name>]
                          [DESCRIPTION             <description>]
                          <Activity Kind Information>
                          [<Access Restriction part>]              // optional
                          [<Transition Restriction part>]
                          [<extended attribute list>]
                     END_ACTIVITY
                     [<Activity List>]

      <activity id>              ::=   <identifier>


// Activity Kind Information

/* The Activity kind information describes how an Activity is executed */

<Activity Kind Information>      ::=   ROUTE          /* for notational purposes only,
                                          no counterpart expected at
                                          execution time                    */
                                     | IMPLEMENTATION    <implementation>
                                       [PERFORMER        <participant assignment>]
                                       [START_MODE       <mode>]
                                       [FINISH_MODE      <mode>]
                                       [PRIORITY         <priority>]
                                      <simulation information>
                                       [ICON             <icon identifier>]
                                       [DOCUMENTATION    <documentation>]

      <implementation>           ::=   NO                          // not supported by Workflow
                                     | APPLICATIONS <generic tool list>      // applications
                                     | WORKFLOW <subflow reference>          // subprocess
                                     | LOOP <loop kind>                              // repetition
                                        CONDITION <loop condition>

      <generic tool list>        ::=   <tool invocation>
                                     |                     // alternative optional
                                      TOOL_LIST <tool list>
                                        [DESCRIPTION      <description>]
                                       [<extended attribute list >]
                                     /* may e.g. describe in vendor-defined form an
                                       execution control, e.g. sequential, parallel  */
                                      END_TOOL_LIST
      <tool invocation>          ::=   <generic tool>
                                       [<parameter map list>]
      <generic tool>             ::=   [TOOL] <generic tool id>
                                     | PROCEDURE <procedure id>              // optional
                                                                 // library built-in or extended
      <tool list>                ::=   <tool invocation>
```

```
                                        [<tool list>]                          // optional

      <subflow reference>          ::=   <execution> <process id> [<parameter map list>]
      <execution>                  ::=   ASYNCHR | SYNCHR

      <loop kind>                  ::=   WHILE | REPEAT_UNTIL
      <loop condition>             ::=   <condition>

      <participant assignment>
                                   ::=   <performer expression>  // performer qualification

      <mode>                       ::=   AUTOMATIC | MANUAL      //with or without user interaction

      <simulation information> ::= [INSTANTIATION     <instantiation>]
                                        [<time estimation>]
                                        [COST              <cost estimation>]
      <instantiation>              ::=   ONCE
                                        [| MULTIPLE]                 // optional
      <cost estimation>            ::=   <string>
```

Restrictions:

(Loop:)

*(**Connection**:) For each Activity having a LOOP implementation there exists exactly one Transition having its* <activity id> in the FROM LOOP part and *one Transition having its* <activity id> in the TO LOOP part (= connecting Transitions).

*(**Blocking**:) For the Transition-Activity Network spanned by the connecting Transitions, i.e. the Activity referenced in their TO part and that in their FROM part (= border Activities), the Blocking Restrictions holds.*

*(**No Waiting**:) Loop Conditions are evaluated directly, i.e. if they are evaluated to FALSE there is no waiting for a change to TRUE.*

*(**Termination**:) If the <transition condition> is never evaluated to TRUE (REPEAT_UNTIL) or FALSE (WHILE), respectively, then this configuration is outside the scope of what the WPDL defines, and the semantics then is not specified by this standard (i.e. is vendor dependent).*

*(**Preliminary End**:) If during execution of the Loop body of a Loop Activity an Activity is reached that is not referenced in the FROM part of any Transition, then the behaviour is the same as if it is not part of a Loop body.*

Explanation:

The restriction describes that - in case of no repetition - the behaviour is the same as if instead of the Loop an Inline Block is used. In this case the part of the Loop body that is not connected by any (regular) Transition path with its Loop-End Transition is executed "asynchronously" until termination, and the Loop-End Transition is not synchronised with this part of the Loop (i.e. no implicit Join).

// Access Restriction part

/ The Data Access Restriction part describes restrictions to the Data accessible by an Activity */*

Restriction:

*(**Restricted Data Access**:) If an Activity has an <Access restriction part>, then only <data id>s in its <parameter list> may be used in the actual parameters of its implementation and in its <performer expression>.*

Explanation:

This <Access Restriction part> ("view") is only a filter, i.e. is does not introduce a new behaviour for parameter passing semantics. In particular it does not invalidate the description of parameter passing semantics related to the data in the data space of a Process and Workflow Model and does not introduce local data for Activities.

// Transition Restriction part

/ The Transition Restriction part describes restrictions to the execution and structure of Transitions adjacent to an Activity */*

```
<Transition Restriction part> ::=
                                  [<Inline Block Information>]          // optional
                                  [JOIN <JOIN characterisation>]>       // JOIN part
                                  [SPLIT <SPLIT characterisation>]      // SPLIT part


<Inline Block Information>        ::=   <begin block>          // first Activity in block
                                         | <end block>  // last Activity in block
      <begin block>               ::=   INLINE_BLOCK_BEGIN <block id>
                                         [NAME        <name>]
                                         [DESCRIPTION<description>]
                                         [ICON              <icon identifier>]
                                         [DOCUMENTATION     <documentation>]
                                         [<extended attribute list>]
                                        END_INLINE_BLOCK_BEGIN
      <end block>                 ::=   INLINE_BLOCK_END <block id>
      <block id>                  ::=   <identifier>

<JOIN characterisation>           ::=   AND | XOR

<SPLIT characterisation>          ::=   AND | XOR <list of Transitions>  // regular Transitions
      <list of Transitions>       ::=   <transition id> [<list of Transitions>]
```

Restrictions:

(Inline Block):

*(**Pairing**:) For each Activity with a <begin block> part (i.e. the block begin) there exists a corresponding Activity with an <end block> part (i.e. the block end) with the same <block id> and vice versa.*

*(**Forward Reachability**:) For each block begin Activity the corresponding block end Activity is reachable by a (forward) thread (i.e. they are not the same).*

*(**Blocking**:) For the Transition-Activity Network between the <begin block> and <end block> activities (= border Activities) the Blocking Restriction holds.*

*(**Blocking Restriction**:) There is a Transition-Activity Network connecting the two border Activities, and this Network is connected to the other parts of the Process Definition only via its Border Activities (i.e. only the (regular) Transitions in this Network may have one of the other Activities in this Network in their FROM or their TO parts and vice versa.*

(JOIN:)

*(**Complete JOIN Covering**:) If an Activity has more that one incoming (regular) transition, then a <JOIN part> is required. If there is a <JOIN part>, then all incoming (regular) Transitions of the present Activity are COVERED by this <JOIN part>).*

*(**XOR Uniqueness**:) An XOR JOIN assumes that the Activity is only reached by one of the specified incoming (regular) Transitions and no waiting is required. If it is reached by more than one incoming (regular) Transitions, then this configurations is outside the scope of what the WPDL defines, and the semantics of the whole process then is not specified by this standard (i.e. is vendor dependent).*

(SPLIT:)

*(**Complete SPLIT Covering:**) If an Activity has more that one outgoing (regular) Transition, then a <SPLIT part> is required. If there is a <SPLIT part>, then all outgoing (regular) Transitions of the present Activity are referenced in this <SPLIT part>.*

*(**OTHERWISE condition evaluation:**) If in an AND SPLIT one of the outgoing Transitions has an OTHERWISE condition,*

then all (regular) Transitions condition(s) except that one are evaluated in parallel. If after evaluation of those none is evaluated to TRUE, then the Transition with the OTHERWISE is followed.

*(**Sequential condition evaluation:**) In an XOR SPLIT the evaluation of Transition conditions of the <list of Transitions> is sequential, from left to right, and the first (regular) Transition for which its condition is evaluated to TRUE is chosen.*

Explanation:

In an XOR SPLIT the <Transition Conditions> of these outgoing (regular) Transitions are evaluated in the sequence given by the order of the <sequence numbers>. If there is an outgoing Transition without <Transition condition>, then this is assumed to be the OTHERWISE, and no other Transition following this one in the order will ever be executed. The Transition with the OTHERWISE condition is treated equivalent to one having condition TRUE (i.e. no different behaviour as in the case of an AND SPLIT).

4.3.4. Transition Information

The Transition Information describes the possible transitions between the activities and the conditions under which they are taken into account. Further control and structure restrictions may be expressed in the Activity definition.

```
// Transition Information

<Transition Information List>::=
                                TRANSITION                  <transition id>
                                      [NAME                 <name>]
                                      <transition kind description>
                                      [DESCRIPTION          <description>]
                                      [<extended attribute list>]
                                END_TRANSITION
                                [<Transition Information List>]

      <transition id>           ::=   <identifier>
      <transition kind description> ::=
                                      FROM <activity id> TO <activity id>
                                        [CONDITION          <transition condition>]
                                                    // regular Transition
                                      | FROM LOOP <Loop activity id> TO <activity id>
                                                    // Loop-Begin Transition;
                                                    // connects first Activity in loop body
                                      | FROM <activity id> TO LOOP <Loop activity id>
                                                    // Loop-End Transition;
                                                    // connects last Activity in loop body
      <transition condition>    ::=   <condition>
                                       | OTHERWISE
      <Loop activity id>        ::=   <activity id>         // implemented by Loop
```

Restrictions:

*(**No Waiting**:) Transition Conditions are evaluated directly, i.e. if they are evaluated to FALSE there is no waiting for a change to TRUE.*

*(**Open Successor**:) If after evaluation of the Transition condition(s) of the outgoing (regular) Transition(s) of an Activity none of the possible successor Activities is reached, then this configurations is outside the scope of what the WPDL defines, and the semantics then is not specified by this standard (i.e. is vendor dependent).*

*(**Pairing**:) For each Transition having a FROM LOOP there exists a corresponding Transition having a TO LOOP referencing the same <Loop activity id> and vice versa.*

*(**EVALUATION Semantics**:) The OTHERWISE condition is evaluated to TRUE but obeys special semantics. Further details and restrictions in the semantics of Transition and condition evaluation are described in the Activity Definition (chapter4.3.3).*

*(**At Most One OTHERWISE**:) The set of outgoing transitions for an Activity may include no more than one Transition with an OTHERWISE condition.*

4.3.5. Workflow Application Declaration

Workflow application declaration is a list of all applications or tools required and invoked by the workflow processes defined within the WPDL-file. A workflow application declaration may have parameter definitions used for the invocation parameters and also used within other entities.

```
<Workflow Application List> ::=
                        APPLICATION                         <generic tool id>
                              [NAME                         <name>]
                              [DESCRIPTION                  <description>]
                              [TOOLNAME                     <tool name>]
                              [<formal parameters>]
                                // invocation interface parameters of application
                              [<extended attribute list>]
                        END_APPLICATION
                        [<Workflow Application List>]

                        <generic tool id>                   ::=    <identifier>
                        <tool name>                         ::=    <string>
```

4.3.6. Workflow Relevant Data

Workflow relevant data represent the variables of a workflow process or workflow model definition.

```
<Workflow Relevant Data List> ::=
                        DATA                                <data id>
                              TYPE                          <complex data type>
                              [NAME                         <name>]
                              [LENGTH                       <cardinal>]
                              [DEFAULT_VALUE                <value>]
                              [DESCRIPTION                  <description>]
                              [<extended attribute list>]
                        END_DATA
                        [<Workflow Relevant Data List>]

      <data id>                     ::=   <identifier>
      <value>                       ::=   <initial>                       // of appropriate type
                                          | <function access>             // of appropriate type
```

4.3.7. Workflow Participants

The Workflow Participants are those elements of an Organisational Model that are either acting parties in a Workflow Process or responsible for it. The definition is an abstraction level between the real performer and the activity which has to be performed. It may refer to an external organisational model. Actors may be defined by a membership in an organisational unit, by a function, role or competence, by relations to actors of already performed activities etc., we call it the type. WPDL supports a basic set of types: organisational unit, human, role, system, relation to process history.

We distinguish a regular Organisational Model definition (called Workflow Participant Definition) describing the OM entities and their types and optionally their relationships to one another as far as they are workflow relevant, and a Minimal Organisation Model Definition (called Workflow Participant Declaration) that is only a list of participant identifiers with an optional type characterisation.

The Workflow Participant Definition is provided in a separate document. It is defined in Organisational Model Definition.

```
<Workflow Participant Specification]
                      ::=   <Workflow Participant Declaration>


<Workflow Participant List>   ::=
                      PARTICIPANT                       <participant id>
                            [NAME                       <name>]
                            [DESCRIPTION                <description>]
                            [<extended attribute list>]   // for participant specific attr.
                            <participant type description>
                      END_PARTICIPANT
                      [<Workflow Participant List>]

      <participant id>             ::=   <identifier>

<participant type description>
                                   ::=   <declaration Ptype description>
                                         | <definition Ptype description>

<Workflow Participant Declaration>
                                   ::=   <Workflow Participant List>
                                           // using <declaration Ptype description>
<declaration Ptype description>
                                   ::=   [TYPE           <Ptype key>]

<Ptype key>                        ::=   <ou key> | <hu key> | <ro key> | < system key>
      <ou key>                     ::=   ORGANISATIONAL_UNIT             // an organisational unit
      <hu key>                     ::=   HUMAN                           // a human
      <ro key>                     ::=   ROLE                            // a role
      <system key>                 ::=   SYSTEM
```

The productions for <Workflow Participant Definition> and <definition Ptype description> are provided in a separate document.

5. Annex A: Conformance

This is a preliminary attempt to define conformance for vendors supporting WPDL. We expect responses from vendors to the Beta document will help fashion the final content of this Annex.

In the first section the overall concepts are sketched. In the second section the control-relevant parts are extracted from the grammar. This allows distinguishing "grammar optional" parts from "conformance-optional" characteristics. The WPDL syntax only contains comments "// optional" relating to the conformance-optional parts.

5.1. General Concept

The following general concepts are proposed:

- ***Essentials***
    - A vendor claiming to be in conformance with the WPDL has to be able to read the *complete WPDL syntax*. (This does not imply that all conformance-optional parts are supported in execution of the WPDL.)
    - A vendor claiming to be in conformance with the WPDL has to export an imported WPDL definition as-is without changes. This also holds for parts not supported by this vendor.
- ***Conformance Profile***
    - A vendor claiming to be in conformance with the WPDL has to publish a Conformance Profile containing all those conformance-optional (conformance-relevant) parts of the WPDL supported.
    - There may be different Conformance Profiles for reading and writing Process definitions.
- ***Extended Import Capability***
    - A vendor may be able to execute WPDL Definitions exported by vendors including conformance-optional parts in the writer's Conformance Profile if these non-supported features are not used in the concrete Process definition (e.g. the use of RESTRICT_TO_IF).
    - There may be special conversions between Conformance Profiles. In later releases these conversion algorithms might be part of an extended WPDL definition (e.g. as an Annex).
- ***Conformance Classes***
    - An aim for the future is to define a limited set of Conformance Classes. These may be deduced from the Conformance Profiles
    - A first approach is described by the Conformance Classes defined in the Workflow Model Definition.

5.2. Conformance-relevant WPDL Elements

This chapter provides an overview about conformance-relevant elements identified during analysis of products or in discussion with vendors. It is not assumed that it is exhaustive.

5.2.1. Overview

We can distinguish three areas where Conformance Profiles have to be taken into account:

- WPDL entities
- Expressions
- Name Spaces

Each of these areas may give rise to different conformance profiles.

5.2.2. Name spaces

The WPDL mostly assumes that name spaces are disjunct. Vendors may want to have further restrictions that cannot be easily handled by the importing program.

5.2.3. Expressions

production left side	***optional contained parts (right side)***	***Alternatives and Comment***
<reference> basic data type> <REFERENCE-T>	<REFERENCE-T> REFERENCE	The basic data type Reference is supported or not supported
(any)	<extended attributes>	The extended attributes of a specific vendor may be supported or not supported

Table 5-1: Conformance Options: Expressions

There may be further restrictions on the use of expressions, e.g. in parameters. There may also be restrictions in the use of Library Functions in Expressions.

5.2.4. Workflow Entity Attributes

5.2.4.1. Workflow Model Attributes

All those Attributes that are optional in the grammar are alternatively supported or not supported except the entity definitions/declarations which have to be supported.

5.2.4.2. Workflow Process Attributes

All those Attributes that are optional in the grammar are alternatively supported or not supported except the entity definitions/declarations which have to be supported.

In addition the following special rules hold:

production left side	*optional contained parts (right side)*	*Alternatives and Comment*
<Workflow Process Definition>	<Workflow Participant Specification>	Required or not required
<Workflow Participant Specification>	<Workflow Participant Definition>	Supported or not supported

Table 5-2: Workflow Conformance Options

5.2.4.3. Workflow Process Activity

The following parts of the WPDL grammar are optional with respect to conformance:

All those Attributes in the <Activity List> (top level production) that are optional in the grammar are alternatively supported or not supported for the non-default values except the PERFORMER attribute, which has to be supported.

The PERFORMER attribute may be required or not required (i.e. set to default).

In the lower levels of the productions the following elements are optional:

production left side	*optional contained parts (right side)*	*Criterion, Comment*	*Alternatives*
<generic tool list>	<single list>	Supported	(Y) (N)
<generic tool list>	<extended flow control>	Supported *(if not supported and single list supported, then the default is sequential)*	(Y) (N)
<generic tool>	<procedure id>	Library procedures are Supported	(Y) (N)
// SPLIT:			
AND	Conditions in Transitions	Supported in the related Transitions	(Y) (N)
AND	<OTHERWISE clause>	Supported	(Y) (N)
XOR	<OTHERWISE clause>	Supported	(Y) (N)
XOR	<OTHERWISE clause>	If supported, then required	(Y) (N)

Table 5-3: Activity Conformance Options

5.2.4.4.Transition Information

See SPLIT/JOIN

5.2.4.5.Workflow Application Declaration

All those Attributes that are optional in the grammar are alternatively supported or not supported for the non-default values except the <formal parameters> attribute, which has to be supported.

5.2.4.6.Workflow Relevant Data

All those Attributes that are optional in the grammar are alternatively supported or not supported for the non-default values except the LENGTH and DEFAULT_VALUE attribute, which have to be supported.

A LENGTH attribute may be required or not required.

5.2.4.7.Organisational Model (Workflow Participant Declaration)

All those Attributes that are optional in the grammar are alternatively supported or not supported.

All productions of <Ptype key> are alternatively supported or not supported. At least one of them has to be supported.

6. Annex B: Aspects of a Formal Semantics

The semantics of the WPDL is defined in an informal way. However, for some parts a more formal description would provide benefits. This annex collects some aspects relevant for it.

6.1. Process Instance States

The following[1] describes a set of standard valid states for each of the major workflow objects defined in this document. A state for a particular workflow object can be identified by its name only or by specifying its full name including its super-state parents using dot notation.

The top level of states for a Process Instance distinguishes two states, *open* and *closed*. The open state has two sub-states, *running* and *notRunning*; notRunning in turn has two sub-states, *notStarted* and *suspended.* The following list describes the states in detail:

- ***open*** - the Process Instance is being enacted
- ***open.running*** - the Process Instance is executing
- ***open.notRunning*** - the Process Instance is temporarily not executing
- ***open.notRunning.notStarted*** - the Process Instance has been created, but was not started yet
- ***open.notRunning.suspended*** - execution of the Process Instance was temporarily suspended
- ***closed*** - enactment of the Process Instance has been finished
- ***closed.aborted*** - enactment of the Process Instance has been aborted by a user (see the specification of WMAbortProcessInstance for a definition of abortion in contrast to termination)
- ***closed.terminated*** - enactment of the Process Instance has been terminated by a user (see the specification of WMTerminateProcessInstance for a definition of termination in contrast to abortion)
- ***closed.completed*** - enactment of the Process Instance has completed normally (i.e., was not forced by a user)

An implementation might decide to support refinement of states to a certain level only or omit certain states; valid sets of states include for example:

- *open* and *closed*
- *notRunning*, *running* and *closed*
- *notStarted*, *running*, *completed* and *terminated*
- ...

The following diagram shows the states and potential state-transitions; transitions are shown for the bottom-level states only, transitions between the higher-level states can be deduced from that easily; e.g., there is a transition from *open* to *closed* or from *notRunning* to *running*, but no transition backwards in both cases.

[1] Cutout of: Document Number WFMC-TC-1009 "Workflow Client Application (Interface 2). Application Programming Interface (WAPI). Specification Version 2.0 (Beta) 01-October-96, Appendix G.

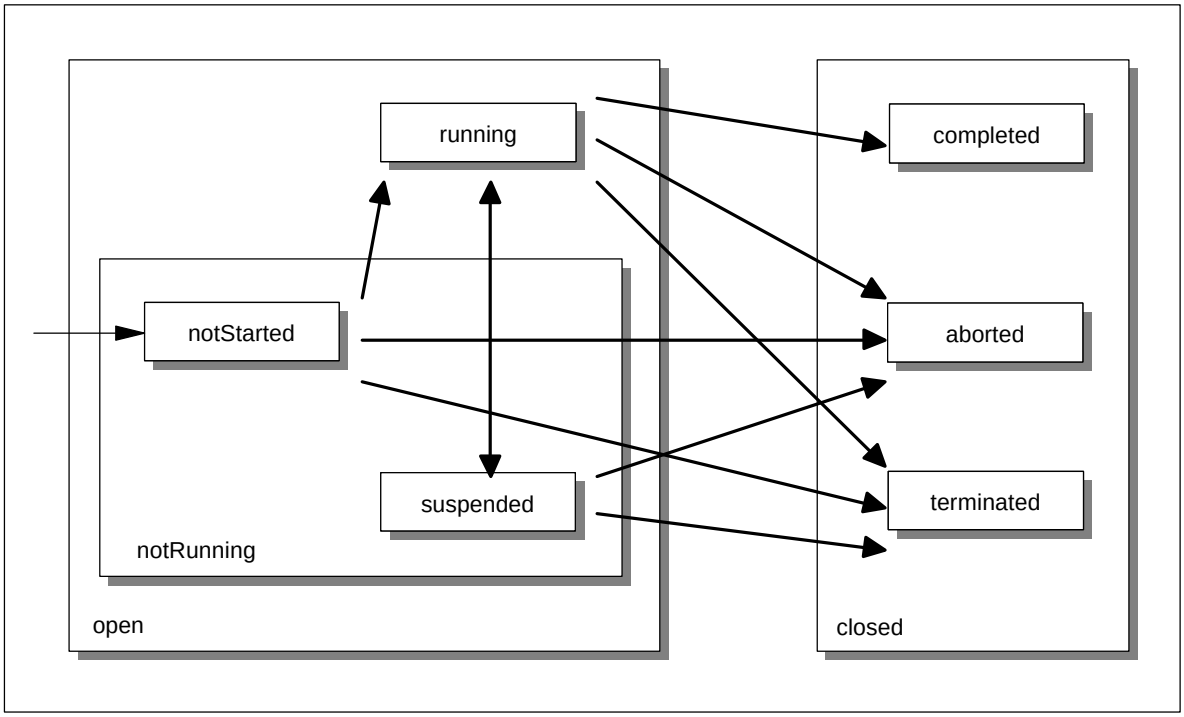

Figure 6-1: Process Instance States

Here is a short discussion of the various state-transitions:

- When a Process Instance is created it will take its intial state, which is *open.notRunning.notStarted* (or just *open*, or *open.notRunning* depending on the level of granularity supported)
- Transitions can be made from *notRunning* states to the *running* state; transitions from the *running* to the *notRunning* super-state can be made to the *suspended* sub-state only.
- When enactment of a Process Instance is finished, its state will take one of the flavours of the *closed* state, depending on the way of ending enactment (normally *completed*, *terminated* or *aborted*). The *completed* state can only be reached from the *running* state since it represents normal completion of the Process Instance; the other *closed* sub-states are reached via the WMAbrtProcessInstance or WMTerminateProcessInstance operations.
- The *closed* state is a final state, i.e., there is no transition from a *closed* state to an *open* state.

7. Annex C: Analysis of PDL's

Preface: The work described in this chapter was performed at the beginning of work on WPDL. Since then the terminology used has changed to some extent. In this chapter the terminology is therefore not always in conformance with the rest of the document.

This recommendation for a common WPDL is based upon a number of vendor specific PDL's. It was seen, that there are different methods to describe the organisational parts of a process:

- a procedural description based on workflow primitives (parallelism, alternative, loop, ...) and stepwise decomposition;
- a directed graph description with pre- and post-conditions for the activity nodes;
- a variety of petri net descriptions;
  - simple petri nets
  - predicate transition nets
  - coloured petri nets
  - funsoft nets
  - etc,
- a description of speech act networks (event driven nets).

The analysis of three different PDL's was undertaken to arrive at a "Minimum Meta Model".

In general one could say, that the discussed PDL's contain more or less the same information. However the following issues arise:

- the bundling of process definitions into sections is done differently (e.g. transition conditions are carried in an extra section rather than the pre- and post-conditions of an activity; data flow is described separately rather than as input/output parameters of an activity)

It was agreed that WPDL should support two philosophies - to have all data described in one flat process definition file and, on the other hand, to allow references between separate files. Nevertheless the main focus is on processes, subprocesses and activity definitions. Organisational models are out of the scope for the first step.

To find a proper recommendation for WPDL we started by taking the Meta Model Entities and looked up the information corresponding to these entities in the three different PDL's. As an example, we were able to break down the major entities as shown in the following table.

Meta Model Entity	**IBM Flowmark Terminology**	**Ley Cosa Terminology**	**SNI WorkParty Terminology**	**Recommended WPDL Term**
Workflow Type Definition	PROCESS ... END	Declare Flow ... End Flow	Module ... End Module	WORKFLOW <workflow_name> : END_WORKFLOW
Workflow Participant	PERSON	(ref.)	(ref.) resp. part of activity	PARTICIPANT : END_PARTICIPANT
Workflow Process Relevant Data	STRUCTURE	(implicitly; defined through usage)	Declarations (InPins, OutPins, Variables) resp. (ref.)	DATA : END_DATA
Transition Information	CONTROL FROM .. TO ..	(in-/out conditions; part of activity) Declare Condition ... End Condition	Control Flow (Alternative, Loop, Parallel, Split, Sequence) (workflow primitives equiv. to and/or join resp. split	TRANSITION FROM ... TO ... : END_TRANSITION
Activity	PROGRAM_ ACTIVITY ... END	Declare Activity ... End Activity	Activities resp. (ref.)	ACTIVITY : END_ACTIVITY
Workflow Application	PROGRAM ... END	(ref.)	part of Activities resp. (ref.)	APPLICATION : END_APPLICATION
- **(data flow)**	DATA FROM	part of declare activity	part of activity	(part of ACTIVITY)

Table 7-1: Evaluating different PDLs

In the first column the table contains the entities of the Minimum Meta Model, the appropriate elements of the considered PDL's are shown in the next three columns and, in the last column, the recommendation for WPDL.

The analysis of the different PDLs brought up the following aspects:

- Transition conditions are partly conditional and partly unconditional. Unconditional transitions are sometimes defined implicitly.
- Some PDL's define conditions in the process layer as transition conditions while others define the conditions as pre- or post-conditions of an activity. Others support both kinds of definition.
- Data flow is also defined differently. Some PDL's define the data flow as input/output parameters (attributes) of an activity (an activity consumes ... and produces ...) while others define data flow separately from the activities.
- Certain PDL's define the organisational aspects as an integrated part of the process definition while others define attributes within the description of the activity, which points at a separate organisational management system.
- Many aspects of a process (not only the transition conditions) depend on runtime evaluations of process relevant data - input/output parameters, workflow participant definitions, etc.
- Some PDL's offer workflow primitives, which are a combination of AND JOIN, AND SPLIT, OR JOIN and OR SPLIT. With these primitives they support alternatives, loops, parallelism, etc. These constructs imply a hierarchical structure of the control flow and therefore a stepwise decomposition whenever the control flow is interpreted. Other PDL's have a flat control flow structure. They describe the control flow by defining all connections in the network of activities on a single level. This implies for all nodes an evaluation of their neighbours. Some of these single level approaches allow loops, others do not, they provide a recursion during runtime (copying one part of a graph to a different place).
- Some PDL’s support the modelling of business cases in one very large network, while others are suited for the definition of larger cases in a couple of networks with synchronisation between them. In this case a mapping from one PDL in the other one via WPDL will be really difficult.

As a result of the evaluation we may fix, that in some cases an automatic translation of a certain vendor specific PDL into another one will cause a big lag in information. In such cases a bidirectional implementation of the PDL exchange is either very expensive or has a bad result. It is obvious, that this situation does not change by making the translation via WPDL.

8. Annex D: WPDL Translators

8.1. Design Principles for WPDL Translators

Design principles for translators:

- Process descriptions should be automatically translated back and forth between WPDL and other process representations with as little loss of meaning as possible. If translations cannot be done fully automatically, the human efforts needed to assist the translation should be minimised.
- If a translator cannot translate a part of a WPDL process description to its target format, it should translate as much of the description as possible (and not, for example, simply issue an error message and give up). Second: It should represent any untranslatable parts in a way that lets a person understand the problem and complete the translation manually if desired. Third: It should preserve any uninterpretable parts so that the translator can add them back to the process description when it is translated back into WPDL.

Most graph-oriented vendors will not have features to describe directly the *SPLIT/JOIN part* of the Transition Information. If a WPDL definition using these features is imported, it might be useful for the vendor to define a corresponding activity type with predefined semantics corresponding to the SPLIT/JOIN semantics of WPDL.

8.2. A LEX and YACC Version for WPDL

The LEX and YACC version are distributed as a separate Document (Document Number WfMC TC-1016-Y). Its purpose is to prove the implementability of the Grammar. In case there is a discrepancy between the BNF-like and the YACC version of WPDL the former is the primary one.

To convert the WPDL BNF-like notation into a YACC notation some changes were required:

- Left recursion had to be changed into right recursion (?? or vice versa ???). This provided no problems.
- Some productions in the WPDL BNF-like notation have been provided in a simplified tabular version. These had to be expanded for the YACC version.
- The following simplifications were made:
    - The Basic REFERENCE has been treated as string.
- To cope with some semantics definition C-code had to be added.

9. Annex E: Document History

October 1995 (Draft 2. 0)
Document including
Metamodel (Process Elements)
Proposed WPDL Grammar
Analysis of PDL's
Representative Business Example description
Concept of mandatory and optional parts

May 1996 (Draft 5.1)
Revisions and Extensions:
Role extended to Participant Hierarchy
Expressions and parameter handling added
Extended Attributes
Editorial enhancements (introduction chapter)
First attempt to provide a Beta (withdrawn due to consistency problems)

October 1996 (Draft 6.9)
Revisions and Extensions:
Expression reformulation
Identifier and names separated; scoping rules defined
Transitions enhancements and reformulated
Separation of Performer (of an Activity) from Participant Hierarchy (Organisation Modelling); Performer expressions for Activity-Performer relationship
Environment access by Library Functions and Procedures
Workflow Model concept instead of Workflow File; External Model Access; external Organisation Model possibility
Editorial enhancements: Informal description of WPDL elements; Overall WPDL appearance

December 1996 (Draft 6.94)
Revisions and Extensions:
Identifiers name space separated from others by including in single quotes
Transitions revised (explicit SPLIT/JOIN part added)
Restrictions of block-structured to Process graphs added (sequential condition evaluation; first approach for inline-block)
First YACC version of WPDL; Feedback: removal inconsistencies in WPDL syntax

March 1997 (Draft 6.94g)
Revisions and Extensions:
Transitions revised
Separation of "Further Extension" parts from kernel, moving the former to Annex
Separate conformance relevant parts in own chapter

October 1997 (Draft 6.95)
Revisions and Extensions:
Transitions and Split/Join concept revised; cascading conditions permitted; route (transition)

concept
Activity Pre- and Post-Conditions removed, integrated into Transition and Split/Join concept
Separation of Participant Declaration (plain role concept) from full Participant Definition (Organisation Model)
Further unclear parts moved to "further extension" Annex (e.g. Abilities)
Metamodel picture revised
Document structure revised: Document split into different parts:
Process modelling (part p)
Organisation Modelling (part o)
Examples enhanced by Q&A part (part x)
YACC version (part y)
Conformance chapter revised (conformance profile element list)

March 1998 (Draft 6.96b)
Revisions and Extensions:
Transitions and Split/Join concept revised; route activity concept introduced
Inline Block introduced
Loop as Implementation of an Activity
Conformance attribute for graph classes added
Restrictions as part of the grammar for restrictions not expressed in the syntax (e.g. context conditions)

July 1998 (Draft 7.0 Beta)
Revisions and Extensions:
Syntax alignment for details
Workflow Process Model enhanced by Workflow Repository concept
Environment access explicitly introduced, Predefined Library elements included
Organisation Model (Participant definition) completely removed to a separate document (Participant declaration part kept)
Further unclear parts moved to "further extension" Annex (e.g. Abilities)
Parameter handling , especially for Process Parameters, made more precise.
External relationships in Process Metamodel diagram added
Document update
Formulation revision (Prepared for Beta)
Further Extension part split into separate document

August 1998 (7.03 Beta)
Keyword RESOURCE changed to keyword SYSTEM in Participant attribute values.
Semantics of instances of workflow relevant data defined on the model level have been made more precise in the WPDL part.
Additional explanations and figures have been added in meta model and informal description part.
For clarification purposes details have been added in the Restrictions of the Activity and Transition definition.

10. Index

10.1. Index of Figures

10.2. Index of Tables