Chapter 25

A DATA AND EVENT ORIENTED WORKFLOW PROCESS DEFINITION METAMODEL COHERENT WITH THE UML PROFILE FOR EDOC SYSTEMS

José A. Soto Mejía

Abstract: This paper presents a plausible mapping between the adopted UML Profile for EDOC systems and one of the submissions to the OMG's Request for Proposal for a Workflow Process Definition Metamodel. Then, given that the proposed by DSTC Workflow Process Definition Metamodel does not consider the specification of events therefore in this paper it is suggested a new workflow process definition metamodel that include these aspects with a clear separation of the data and event oriented control flow dependencies and that is compatible with the UML Profile for EDOC systems. Furthermore, in order to build the proposed in this paper workflow definition metamodel as an UML profile, the new workflow metamodel is derived (stereotyped) from the UML 1.4 Activity Graphs. To introduce a representation that be computationally interpretable by a workflow engine a textual notation is introduced to represent a workflow process definition coherent with the proposed Workflow Process Definition Metamodel. As a proof of concept a simple prototype of a workflow engine able to interpret the textual representation of the workflow process definition was implemented in the Java language.

Key words: Workflow, Process Definition, Metamodels, Enterprise Distributed Object Computing, EDOC, UML profile.

1. INTRODUCTION

The adopted by the Object Management Group (OMG), UML Profile for Enterprise Distributed Object Computing (Heaton, 2002), supports the requirements for driving an object oriented design of an enterprise computing system (a software system that provides support for carrying out an integrated set of business processes across an enterprise) to an implementation in an enterprise distributed computing environment using an enterprise class component model.

It is recognized that a successful implementation of such a system requires the operation of the system to be directly related to the business processes it supports. As it is also recognized that workflow management systems are today essential to corporate organizations that need to automate their business processes since they allow organizations to specify, execute and efficiently monitor their business processes. The specification of those parts of the business process to be automated is done using the model elements of a given workflow process definition metamodel or its corresponding workflow process definition language. In this sense, a few years ago (December 2000) the OMG started the process (Boldt, 2000) to standardize a metamodel and/or a profile which extends the UML for defining workflow processes and more recently the OMG has issued a more general Request for Proposal-RFP- (Cohete, 2003) that solicits submissions that specify a business process definition metamodel, which is platform independent with respect to specific business process definition languages.

To avoid a frequently misunderstanding between business analysts (workflow modelers) and software developers it is important that a clear mapping exists between the workflow process definition metamodel and the enterprise computing system metamodel. With this in mind, in section 2, this paper presents a plausible mapping between the above cited UML Profile for Enterprise Distributed Object Computing (EDOC Profile) and the submitted by DSTC (The Cooperative Research Center for Enterprise Distributed Systems Technology) response (Keaton, 2001) to the OMG's Request for a Proposal for Workflow Process Definition Metamodel (Boldt, 2000).

In Section 3, given that the proposed by DSTC Workflow Process Definition Metamodel does not consider the specification of events then in this paper it is suggested a workflow process definition metamodel that include these aspects (using the sub-profile for events that is part of the EDOC profile), with a clear separation of the data and event oriented control flow dependencies. Furthermore, in order to build the proposed in this paper workflow definition metamodel as an UML profile, the proposed metamodel is derived (stereotyped) from the UML 1.4 Activity Graphs (Heaton, 2001) and the implications of this derivation for the mapped metamodels (EDOC Profile and the submitted by DSTC metamodel) are pointed out. To introduce a notation that is computationally interpretable by a workflow engine, Section 4 introduces sketches of a notation to represent a workflow process definition coherent with the proposed workflow process definition metamodel. The paper ends with a Conclusions and On-going Work Section 5, that sketches the current state of the standardization process going on in the OMG which goes along the approach proposed in this paper

2. MAPPING BETWEEN METAMODELS

Before the mapping between the proposed in this paper workflow definition metamodel and the EDOC metamodel (the UML Profile for Enterprise Distributed Object Computing) is presented, a brief description of the main meta-entities of each of the metamodels is given in Tables 1 and 2. In Table 1, there is a summary of the main concepts related with the business and events profile for EDOC systems and in Table 2 a summary with the main concepts for the DSTC's workflow process definition metamodel. In order to facilitate the comparison and the identification of the source of context, the EDOC terms are presented in **bold** and the DSTC term in *italic.*

Table 1. Main Concepts from the EDOC Profile

ProcessComponent: It represents an active processing unit-"it does something". It defines a set of **Ports** (a **port** defines a point of interaction between **ProcessComponents**) for interaction with other **ProcessComponents** and it has a set of properties that are used to configure the **ProcessComponent** when it is used.
CompoundTask: It defines how to coordinate a set of related **Activities** that, in combination, perform some larger scale activity, ultimately in the context of a **BusinessProcess**. It is also a container of **Activities**, **DataFlows** between these **Activities**, and the **ProcessRoles** which model bindings to objects required by these **Activities**.
Activity: It represents the execution of a part of a **BusinessProcess** using one of two mechanisms: (i) The creation of a composition of nested **Activities, ProcessRoles** and **DataFlows** described by the **CompoundTask** that the **Activity** references through its uses association. (ii) The execution of some feature of an object bound to a **ProcessRole** instance referred to via the Activity's **performedBy** association. An **Activity**'s PortUsages representing **InputGroups**, which contain **ProcessPortConnectors** representing **ProcessFlowPorts**, are the alternative means by which the **Activity** may supply data to these mechanisms to initiate some action.
DataFlow: It represents a causal relationship in a business process. The source of the **DataFlow** must "happen" before the sink of the **DataFlow**. **DataFlows** also propagate data values between causally related **ProcessPortConnectors**.
ProcessFlowPort: It represents the formal types of inputs to and outputs in the context of a **CompoundTask**. **ProcessPortConnectors** represent the usage of the **ProcessFlowPorts** of a **CompoundTask.**
ProcessPortConnector: It represents the usage of a **ProcessFlowPort** of a **CompoundTask**. A **ProcessPortConnector** may be associated with a source or a sink of a **DataFlow. DataFlows** allow the connection of the **ProcessPortConnectors**, which are owned by a **CompoundTask**.
BusinessProcess: It represents the complete process specification. A **BusinessProcess** defines the **ProcessComponent** view of a process definition that coordinates a set of related **Activities**. It defines a complete business process, which can be invoked, usually using **Ports**, which are connected via **DataFlows** to the **ProcessPortConnectors** of the **Activities**, which it contains. In other words a **BusinessProcess** is an ordinary **ProcessComponent** on the outside, and a **CompoundTask** on the inside.
ProcessMultiPort: It represents a set of related **ProcessFlowPorts** used to describe the inputs and outputs of **CompoundTasks**. They act as a form of correlator for **DataFlows**.
InputGroup: It is a specialization of **ProcessMultiPort**. It is a container for a number of **ProcessFlowPorts**, which are the inputs to a **CompoundTask**, and acts as a form of correlator for

DataFlows.
OutputGroup: It is a specialization of **ProcessMultiPort.** It is a container for a number of **ProcessFlowPorts,** which are the outputs of a **CompoundTask,** and acts as a form of correlator for **DataFlows.**
ExceptionGroup: It represents the outcome of a **CompoundTask** that failed to complete its function.
ProcessRole: It defines a placeholder for concrete **ProcessComponents** that perform an **Activity** or that are used in the performing of an **Activity**. The owner of a **ProcessRole** is a **CompoundTask** and the behavior of the **ProcessRole** becomes part of the behavior of **Activities** to which it is associated.
Publisher: It is a component that exposes a list of **Publications**, and produces **PubSubNotices** accordingly.
Subscriber: It is a role or component that exposes a list of **Subscriptions**, and consumes **PubSubNotices** accordingly.
Publication: It is a declaration of capability and intent to produce a **PubSubNotice**. A Publisher owns it.
Subscription: It is the expression of interest in receiving and capability to receive a **PubSubNotice**. A Subscriber owns it.
EventBasedProcess: It is a **Subscriber** and has **NotificationRules** associated with its **Subscriptions**. It is a **Publisher** and publishes **ProcessEvents**. **ProcessEvents** describe the life cycle of the **EventBasedProcess.**
NotificationRule: It is a rule associated with a **subscription** which determines what should happen within the **EventBasedProcess** holding the **Subscription** when a qualifying **PubSubNotice** is delivered. Optionally an **EventCondition** that requires the delivery of additional events can further guard the **NotificationRule**. A **NotificationRule** is owned by an **EventBasedProcess**
PubSubNotice: It is any data structure that is *announcedBy* a **Publication** and/or *subscribedTo* by a **Subscription**. Instances of **PubSubNotice** are communicated as **DataFlows** from **Publishers** to **Subscribers** based on the **Subscriptions.**
EventNotice: It is any **PubSubNotice** that is triggered by a business event. An **EventNotice** may describe at most one business event.
BusinessEvent: It is any event of business interest that happens within an enterprise. A business event triggers one or more event notices and is described by one or more event notices.
NotificationRule: It is a rule associated with a **Subscription** which determines what should happen within an **EventBasedProcess** holding a **Subscription** when a qualifying **PubSubNotice** is delivered
EventCondition: It identifies a **Subscription** and specifies a **PubSubNotice** instance subset of which one must have been received to satisfy this condition. A **NotificationRule** owns it.

Next Table 2, presents a synthesis with the main concepts that are part of the proposed by the DSTC Workflow Process Definition Metamodel.

Following, the particular characteristics of the mapping between the meta-concepts of the DSTC Workflow Process Definition Metamodel and the EDOC metamodel are explained and Table 3 presents a synthesis of the mentioned mapping.

Table 2. Main Concepts from the DSTC Workflow Definition Metamodel

BusinessProcess: It establishes a context within which sets of business actions, taking place in a prescribed manner, **are** coordinated to achieve some enterprise objective.
Task: It represents a self-contained unit of work. It contains its *InputGroups*, *OutputGroups* and indirectly *Inputs* and **Outputs**.
CompoundTask: It defines how to coordinate a set of related *Activities* that, in combination, perform some larger scale activity, **ultimately** in the context of a *BusinessProcess*.
Activity: It represents the **execution** of a part of a *BusinessProcess* using one of two mechanisms: (i) The creation of an instance of a *CompoundTask* referred to via the Activity's *definedBy* association or (ii) The execution of some feature of an Object bound to a *BPRole* instance via the Activity's *performedBy* association. The *InputGroups* contained by an *Activity* represent the alternative means by which the *Activity* may supply data to these mechanisms to initiate some action.
DataFlow: It represents a causal relationship in a *business process*. The source of the *DataFlow* must happen before the sink of it. They also propagate data values between causally related *DataElements*.
DataElement (abstract): It **represents** data used in *Task*'s input/output. It has two subtypes *Input* and *Output*. A *DataElement* is contained in a *DataGroup*.
Input: It models a collection of data values of a particular type consumed by a *Task* instance. An *Input* is contained in an *InputGroup*.
Output: It models a collection of data **values** of a particular type produced by a *Task* instance. An *Output* is contained in an *OutputGroup*.
DataGroup (abstract): It represents a set of related *DataElements* used to describe the inputs and outputs of a *Task*. They act as a **form** of correlator for *DataFlows*. *InputGroup* and *OutputGroup* are concrete subtypes of *DataGroup*.
InputGroup: It models a set of data values required by a *Task* to do some work. When owned by an *Activity* the *InputGroup* models the actual parameters to some behavior of the *Task*. When owned by a *CompoundTask* the *InputGroup* models the formal parameter to some behavior of the *Task*.
OutputGroup: It represents a possible outcome of the execution of a *Task*. It provides data values associated with that outcome. An **OutputGroup** is contained by a *Task*.
ExceptionGroup: It represents the outcome of a *Task* that failed to complete its function.
BRRole: It defines a placeholder for entities **that** perform an Activity or are used in the performing of an Activity. Its 'find' attributes constraints the set of entities that may be bound to the BPRole at run time.

In the profile for EDOC the EDOC's **BusinessProcess** meta entity inherits from the EDOC's **ProcessComponent** meta entity, which defines a set of **Ports** for interaction with other **ProcessComponents** and that has a set of properties that are used to configure the **ProcessComponent** when it is used (see definitions of the mentioned meta-entities in Table 1). Since in the approved by the OMG profile for EDOC systems, EDOC's **CompoundTask** meta-class specializes EDOC's **BusinessProcess** meta-class the former one (EDOC's **CompoundTask** meta-class) inherits the whole semantics associated with the last one (EDOC's **BusinessProcess** meta-class). Given the above mentioned specific semantic inheritance and being conceptually strict a direct mapping (shown in Table 3) among the meta concepts from the two models (The **CompundTask** meta class from the EDOC's profile and the *CompoundTask* form the DSTC workflow metamodel) should not be plausible.

But, knowing that the definition of a workflow process does not require the details involved with the design of an EDOC system using an enterprise class model we could, just to allow a comparison between the meta entities of the two models, leave temporarily aside the specific EDOC semantic, related with the component design issues, and make the following mapping shown in Table 3.

Table 3. Mapping between the profile for EDOC and the DSTC's Workflow Process Definition Metamodel

Meta-concepts form the Profile for EDOC	DSTC's Workflow Process Definition Metamodel
BusinessProcess	BusinessProcess
CompoundTask	CompoundTask
Activity	Activity
DataFlow	DataFlow
ProcessFlowPort	DataElement
input ProcessPortConnector	Input
output ProcessPortConnector	Output
ProcessMultiPort	DataGroup
InputGroup	InputGroup
OutputGroup	OutputGroup
ExceptionGroup	ExceptionGroup
ProcessRole	BRRole

3. WORKFLOW PROCESS DEFINITION METAMODEL

Given that the proposed by DSTC Workflow Process Definition Metamodel does not consider the specification of events then in this section it is suggested a new workflow process definition metamodel adding these aspects (using the sub-profile for events that is part of the EDOC profile). The new suggested workflow definition metamodel (illustrated in Figure 1) shows a clear separation of the data and event oriented control flow dependencies allowing the specifications of both aspects.

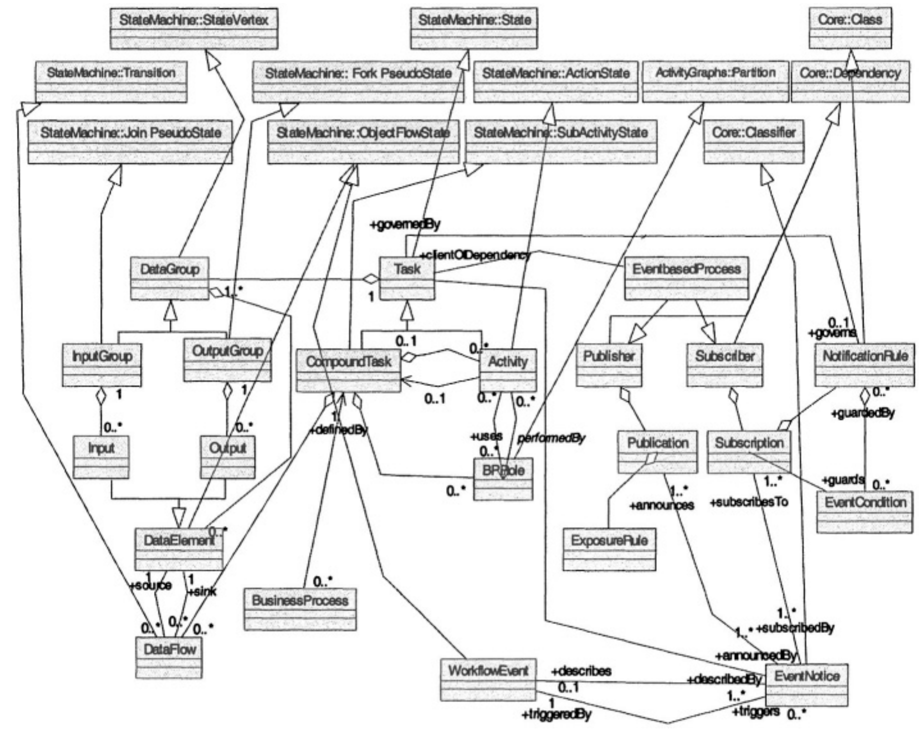


Figure 1. A workflow process definition metamodel with data and event oriented control flow dependencies

In order to derive (stereotype) the proposed in this paper workflow definition metamodel as an UML profile the metamodel is derived from the UML 1.4 Activity Graphs (Heaton, 2001). The use of the UML Activity Graphs semantics to profile the illustrated in Figure 1 metamodel creates difficulties to link an Activity (in the proposed metamodel) with its potential realization. The association of an Activity with the BPRole entity via UML partitions (as in Figure 1) compels us to redefine the DSTC's *BPRole* semantics. The DSTC's *BPRole* has a *type* and an attribute '*find*' to constraint the set of entities to be bound by the *BPRole* at run time. Since, BPRole is a UML's partition and not anymore a class (as it is DSTC's *BPRole*) it is not now possible to associate with it (with BPRole) a *type* and an attribute '*find*' to constraint the set of entities to be bound by the *BPRole* at run time.

If we wanted to be partially coherent with the EDOC metamodel then ProcessRole should subtype an UML meta class. But in this last case (ProcessRole) as a subtype of an UML meta class) it would not be possible to associate in the metamodel depicted in Figure 1 the ProcessRole entity with an Activity, since Activity is being modeled there (see Figure 1) as an UML Action State. If we wanted to be totally compatible with the EDOC profile and used the UML Collaboration as the base to profile the metamodel (as it is the DSTC's approach) then we would have to give up the well accepted way to model business processes using activity graphs.

Since the UML Dependency can associate any two UML Model Elements, then to add to the proposed workflow process definition metamodel all the semantic richness of the EDOC metamodel for Events, a Task (in Figure 1) is associated with an EventBasedProcess that stereotypes UML Dependency.

The BusinessEvent entity from EDOC profile, as defined in Table 1, has been replaced in the metamodel illustrated in Figure 1 by the WorkflowEvent entity with a redefined semantics as follows.

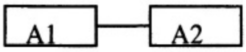
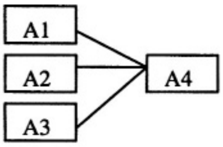
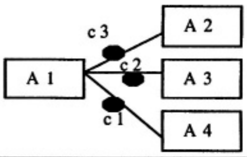
A workflow event is defined in (Hollingsworth, 1999) as the occurrence of a particular situation or condition which has significance to one or more workflows and causes a defined action via the workflow management software. The purpose of this concept is to provide a mechanism, which may be used to co-ordinate or synchronize different processing activities.

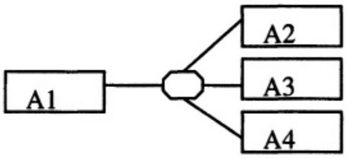
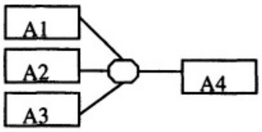
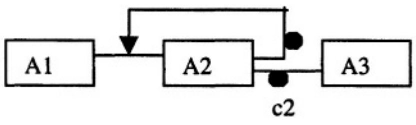
A workflow event has two elements (i) a workflow event trigger or cause and (ii) a workflow event action or response. The workflow event trigger is defined as the set of predefined circumstances that causes a particular action to be taken. The workflow event trigger concept, (as the set of predefined conditions under which the workflow event is emitted), is introduced in the metamodel (see Figure 1) through the meta entity ExposureRule.

The workflow event action is defined as the pre-defined response following the trigger condition. These actions may be associated with specific control actions, to be undertaken by the workflow management software. For example, this might result in the workflow management software enabling a particular activity instance to be started.

The set of control actions and associated conditions is introduced in the model through the meta entity NotificationRules (from the EDOC'NotificationRules). To further specify the control actions and associated conditions it is suggested to use behavioral Rules as the ones described in (CIMOSA). These rules have a clear mapping to the six different Workflow Management Coalition routing possibilities among activities (WfMC, 1999). Table 4 below shows in the first column the CIMOSA behavioral rules and in the second the corresponding routing connection with a diagram sample of each one of them.

Table 4. Mapping between CIMOSA behavioral rules and WfMC routing possibilities

CIMOSA Behavioral Rules	WfMC-Routing possibilities
Process triggering rules: they are used to start a process with or without events (without events means that the business process EF1 is just called by a parent process). WHEN (START WITH *event*-1) DO EF1 WHEN (START WITH *event-1* AND *event*-2) DO EF1 WHEN (START) DO EF1	Start of the process.
Sequential rules: they are used to represent branching conditions in the flow of control. ES(EFi) is a pre-defined function returning the ending status (ES) after completion of EFi, ANY is a reserved word representing all the possible ending status of EFi, and end-status-n represents a specific ending status of EFi. WHEN (ES(EF1) = ANY) DO EF2 (also called forced sequential rule) WHEN (ES(EF1) = *end-status*-1) DO EF2 (also called conditional sequential rule)	**single thread:** an activity A2 can be executed after the completion of another A1.  **or-join:** a point within the workflow where two or more alternative activity workflow branches re-converge to a single common activity as the next step within the workflow.  **or-split:** a point within the workflow where a single thread of control makes a decision upon which branch to take when encountered with multiple workflow branches. The decision is specified by transition conditions (e.g. C1, C2 and C3). 

***Spawning rules*:** they are used do represent the parallel execution of EFi; '&' is the parallel operator, and SYNC is a reserved word meaning the simultaneous start of EF2, EF3 and EF4: WHEN (ES(EF1) = *end-status*-1) DO EF2 & EF3 & EF4 (also called asynchronous spawning) WHEN (ES(EF1) = *end-status*-1) DO SYNC(EF2 & EF3 & EF4) (also called synchronous spawning)	**and-split:** a single thread of control splits into two or more threads, which are executed in parallel within the workflow, allowing multiple activities to be executed simultaneously. 
***Rendezvous rules*:** they are used to synchronize the end of spawning rules: WHEN (ES(EF1) = *end-status-1* AND ES(EF2) = *end- status-2* AND ES(EF3) = *end-status*-3) DO EF4	**and-join:** a point in the workflow where two or more parallel executing activities converge into a single common thread of control. It is a synchronization point in the workflow. 
***Loop rules*:** they are used to execute the same EFi iteratively: WHEN (ES(EF1) = *loop-value*) DO EF1 WHEN (ES(EF1) = *exit-value*) DO Efx	**Iteration:** a workflow activity cycle involving the repetitive execution of one (or more) workflow activity(s) until a condition is met. C1 and C2 are transition conditions. 
***Process end rules*:** they are used to indicate the end of the process: WHEN (ES(EF1) = *end-status*-1) DO FINISH	End of the process

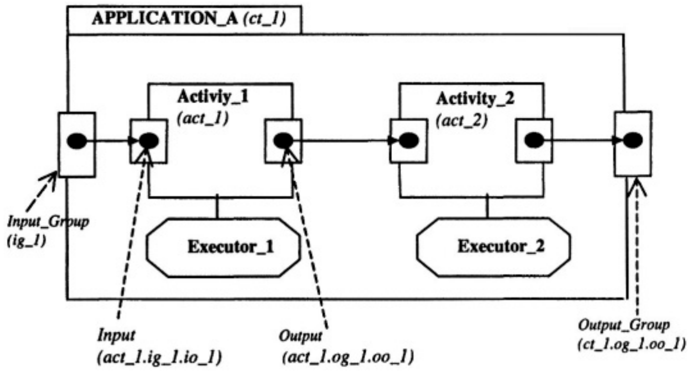


Figure 2. A process definition using an EDOC compatible notation

4. THE TEXTUAL NOTATION

To introduce a notation that be computationally interpretable by a workflow engine, this section introduces sketches of a textual notation to represent a workflow process definition coherent with the proposed workflow process definition metamodel illustrated in Figure 1. As a matter of example, lets assume we have the application of Figure 2, depicted using a compatible EDOC notation. The application is made up of two simple activities linked in strict sequence order and executed by Executor_1 and Executor_2 entities respectively.

The graphical representation shown in the above Figure 2, can be translated into the sketched notation included in Table 5. This notation is to be computationally interpretable by a workflow engine.

The interpretation of this notation is the following. The whole *"Application_A"* is mapped as the content of a *Compound_Task('identifier', 'process_name')* expression. The key word *Process_Defintion()* is used to describe each one of the activities that conform the application as a whole. Inside this *Process_Definition()* section, the constituent process steps are specified using the parameters of the *Activity* (*'identifier', 'step_name', 'BPRole'*) key word expression.

Each one separately, the *Compound_Task(..)* and the *Activity(..)* expressions, must specify its *input* and *output* information using the key constructs *Input_Group*(*'identifier'*) and *Output_Group*(*' identifier'*), and inside each of them, the actual data to be exchanged between the process steps is specified with

input(*'input_object_identifier', 'type')* and *Output('output_object_identifier', 'type')* constructs.

Table 5. Sketch of a Textual Notation

```
Compound_Task( ct_1, Application_A){
 Input_Group( ig_1) { Input( io_1, 'd' ); }
 Output_Group( og_1 ) { Output( oo_1, 'd' ) ; }
Process_Definition() {
  Activity( act_1, Activity_1, Executor_1 ) {
    Input_Group( ig_1 ) { Input( io_1, 'd' ); }
    Output_Group( og_1 ) { Output( oo_1, 'd' ); }
  }
  Activity( act_2, Activity_2, Executor_2 ) {
    Input_Group( ig_1 ) { Input( io_1, 'd' ); }
    Output_Group( og_1 ) { Output( oo_1, 'd' ); }
  }
Dependency (){
    dep( 1, ct_1.ig_1.io_1, act_1.ig_1.io_1 );
    dep( 2, act_1.og_1.oo_1, act_2.ig_1.io_1 );
        dep( 3, act_2.og_1.oo_1, ct_1.og_1.oo_1 );
}
      }
      ,
```

The section *Dependency ()* is used to describe the dependencies among all steps that conform the workflow process. Each individual dependency is specified using the expression *dep(..)* which have three arguments: an identifier of the dependency, the identifier of the source of the information and the identification of the target of the information.

In case of an event oriented workflow process, the event oriented control dependencies could be declared with a syntax similar to the one used when declaring *Input_Group* and *Output_Group*. Following it is a simplified sketch (see Table 6) of the key expressions to declare the intention to produce or to consume workflow events.

A workflow entity specifies its intention to receive a workflow event of type *'event_type'* through the key word *Subscriber(..)* and inside the *NotificationRule* key word are to be specified the associated conditions and control actions to be taken (following the suggested CIMOSA rules as mentioned in section 3.

Table 6. Sketch of a Textual Notation

```
            Subscriber( event_type) {
      NotificationRule ( id, 'CIMOSA_rules');}

            Publisher( 'event_type') {
      EventNotice ('reference_to_event_data');
   RegraDeExposição ( id, 'trigger_conditions');}
```

5. CONCLUSIONS AND STANDARDIZATION WORK

This paper illustrates that there is a clear mapping between the UML profile for EDOC and the DSTC's workflow process definition metamodel that helps to avoid the misunderstanding between business analysts (workflow modelers) and software developers. At the same time that it is possible to build from the DSTC's workflow process definition metamodel, a workflow definition metamodel with a clear separation of the data and event oriented control flow dependencies consistent with the UML profile for EDOC systems. And that the CIMOSA behavioral rules are a plausible way to orient the work to further specify the event oriented control dependencies in a workflow process definition based in the suggested workflow definition metamodel.

Furthermore, the approach proposed in this paper is coherent with the current process that goes on in the Object Management Group (OMG) to standardize the specification of a process definition. The OMG has issued (January 2003) a Request for Proposal (Cohete, 2003) that solicits submissions that specify a business process definition metamodel, which is platform independent with respect to specific business process definition languages. This metamodel will define an abstract language for specification of executable business processes that execute within an enterprise (with or without human involvement) and may collaborate between otherwise-independent business processes executing in different business units or enterprises.

The specification developed in response to that RFP (Cohete, 2003) is expected to achieve the following:

A common metamodel to unify the diverse business process definition graphical and textual notations that exist in the industry.

A metamodel that complements existing UML metamodels so that business processes specifications can be part of complete system specifications to assure consistency and completeness

The ability to integrate process models for workflow management processes, automated business processes, and collaborations between business units.

Adoption of this specification is expected to improve communication between modelers, including between business and software modelers, provide flexible selection of tools and execution environments, and promote the development of more specialized tools for the analysis and design of processes.

With reference to the behavioral rules as the CIMOSA ones mentioned in the above section 3 of this paper there is no generally accepted approach for defining or representing business rules (Hendryx, 2002b). In this sense the work on modeling business rules is in the early stages. The OMG has issued (September 2002) a Request for Information (RFI) for Business Rules in Models (Hendryx, 2002a) and more recently (June 2003) a Request for Proposals (RFP) for Business Semantics of Business Rules (Hendryx, 2003).These works and the relationship of business rules to business processes should be considered in the development of a business process definition metamodel.

REFERENCES

Boldt, Juergen., (2000). "UML Extensions for Workflow Process Definition. Request for Proposal", OMG Document Number: bom/2000-12-11. Available at: ftp://ftp.omg.org/pub/docs/bom/00-12-11.pdf

CIMOSA. "A Primer on key concepts, purpose and business values". Available at: http://cimosa.cnt.pl/Docs/Primer/primer0.htm.

Cohete, Manfred., (2003). "Business Process Definition Metamodel Request for Proposal," OMG Document: bei/2003-01-06. Available at: ftp://ftp.omg.org/pub/docs/bei/03-01-06.pdf

Duddy, Keaton. (2001). "Workflow Process Definition Metamodel submitted by The Cooperative Research Center for Enterprise Distributed Systems Technology (DSTC)", OMG Document Number: cem/2001-12-01. Available at: ftp://ftp.omg.org/pub/docs/cem/01-12-01.pdf

Heaton, Linda., (2001). "Unified Modeling Language Specification" v1.4, OMG Document Number: formal/01-09-67 to formal/01-09-80. Available at ftp://ftp.omg.org/pub/docs/formal/01-09-67.pdf

Heaton, Linda., (2002). "UML Profile for Enterprise Distributed Object Computing", OMG Document Number: ptc/2002-02-05. Available at: ftp://ftp.omg.org/pub/docs/ptc/02-02-05.pdf

Hendryx, Stan., (2002a). "Business Rules in Models Request for Information". OMG Document Number: ad/2002-09-13. Available at: ftp://ftp.omg.org/pub/docs/ad/02-09-13.pdf

Hendryx, Stan., (2002b). "Defining Business Rules. What are they really". OMG Document Number: ad/2002-10-03. Available at: ftp://ftp.omg.org/pub/docs/ad/02-10-03.pdf

Hendryx, Stan., (2003). "Business Semantics of Business Rules Request for Proposal". OMG Document Number: br/2003-06-03. Available at: ftp://ftp.omg.org/pub/docs/br/03-06-03.pdf

Hollingsworth, David., (1999). "Workflow Management Coalition White Paper-Events". Available at: http://www.aiim.org/wfmc/standards/docs

OMG, "The Object Management Group". [http://www.omg.org]

WfMC, (1999). "Workflow Management Coalition. Terminology and Glossary". Document Number WfMC-TC-1011, v3.0 (Feb 99).