A Framework for Unifying Problem-Solving Knowledge and Workflow Modeling

Juan C. Vidal and **Manuel Lama** and **Alberto Bugarín**
Departament of Electronics and Computer Science
Facultad de Física, Edificio Monte da Condesa
15782 Santiago de Compostela, Spain

Abstract

Usually a workflow is described from its control structure and its participants but without taking into account the knowledge used to execute it. This paper outlines a new framework for workflows specification which extends the Unified Problem-solving Method description Language and approaches workflows design from a knowledge perspective. Within this framework, the resources and the control flows that define the traditional workflow framework are defined as knowledge components and are enriched with a knowledge dimension that deals with the definition of the knowledge that is used in its modeling, resource assignments and execution.

Introduction

Workflows are increasingly being used to model processes because they facilitate the communication, coordination and collaboration between the participants of the business process. In last years, a number of specifications have arose for describing processes or workflows such as BPEL4WS (Andrews *et al.* 2003), BPMN (Obj 2006), or the XPDL standard defined by the Workflow Management Coalition (WfMC) (Wor 2005). Most of these languages specify a workflow on the basis of (i) its control structure where the execution control of the activities is defined and (ii) its participants, that specify which agents execute the workflow activities. However, these approaches do not incorporate *explicitly* the problem-solving knowledge in the workflow definition: this knowledge is implicitly used its control structure and in its organizational structure, but as it is not explicitly represented, it cannot be shared or reused. For dealing with this drawback, workflows need to be specified as a new component at the knowledge-level (Newell 1982).

Since the introduction of the knowledge-level concept, several approaches have been proposed to represent knowledge components and, more specifically, problem-solving knowledge components. Between these approaches the Unified Problem-solving Method description Language (UPML) (Fensel *et al.* 2003) can be considered as the *de facto* standard. The main drawback of UPML to be applicable for solving a given problem is that it does not define an operational description for those methods that are not simple and decompose into subtasks: it assumes that this description is at the symbolic level. Based on UPML and from the perspective of semantic web services, there are some approaches, such as IRS-III (Cabral *et al.* 2006) and WSMO (de Bruijn *et al.* 2005), that propose an operational description for the methods. These proposals based on the UPML specification, however, do not use workflows for modeling the execution coordination of the tasks that compose a method (or a service).

In this paper we describe a framework which captures the problem-solving knowledge used to define and execute a workflow. This framework extends the UPML specification by incorporating both the control structure and the participants of a workflow as *two new knowledge components*. The structure of these new components is based on two ontologies: the High-Level Petri Nets ontology (Vidal, Lama, & Bugarín 2006a) and the TOVE ontology (Fox & Gruninger 1998) for process representation and organization modeling, respectively. Figure 1 depicts the stack of the ontologies that describes semantically the different dimensions of a workflow. In this paper when we refer to a workflow, we will take all these dimensions into consideration.

Workflows Specification Framework

Framework architecture, depicted in Figure 2, is composed by a set of boxes, which represent knowledge components, that are connected by a set of ellipses, which represent bridges. Specifically, we define six knowledge components that capture each one of the aspects of a workflow. The first four components, *ontologies*, *tasks*, *methods* and *domain models* are captured by the UPML model, and describe

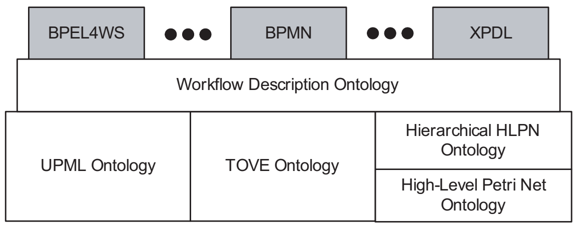
BPEL4WS
BPMN
XPDL
Workflow Description Ontology
UPML Ontology
TOVE Ontology
Hierarchical HLPN Ontology
High-Level Petri Net Ontology

Figure 1: Ontology stack for the representation of workflows

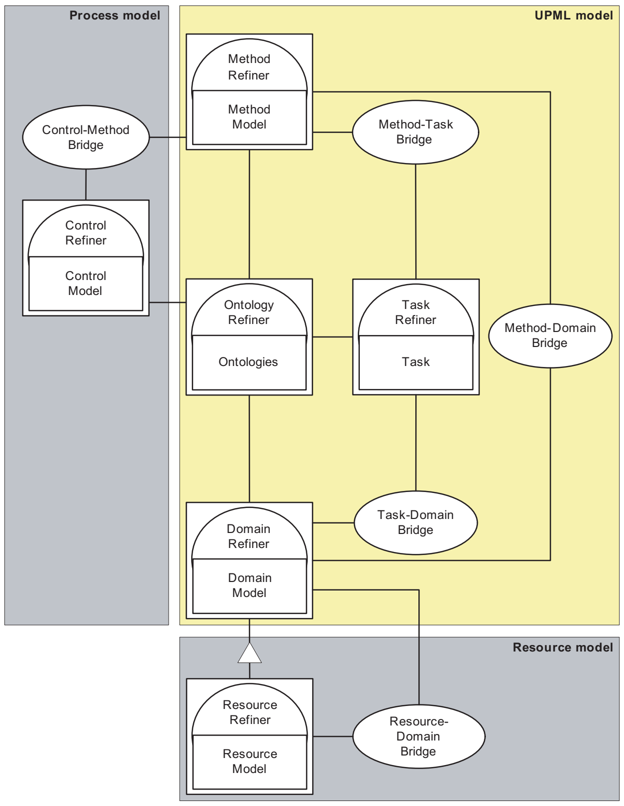

Figure 2: Knowledge-based Workflow Framework

the concept of problem-solving method. The last two components, the *resource models* and the *control models*, complements the problem-solving method specification to cover the definition of the workflow participants and the workflow control structure. *Bridges* define the adapters to connect the knowledge components. For example, bridges are used to relate the control flow to a method, the participants of a workflow or the domain to which the problem-solving method will be applied.

UPML Model

The core of the modeling of problem-solving methods is defined in the UPML model (see Figure 2). Although UPML model is out of the scope of this work, it describes problem-solving methods by means of the following components:

- *Task*. A *task* describes an operation, specifying the required input and output parameters and the pre- and post-conditions. We must remark a task only defines an operation and not the way in which this operation will be solved nor the resources that will participate in its solution.
- *Method*. A *method* details the reasoning process to achieve a task. Like tasks, they specify the required input and output parameters and the pre- and postconditions. However, methods can be split in two types depending on their structure: *non-composite* and *composite* methods. On the one hand, non-composite methods define a reasoning step that cannot be decomposed in more primitive steps. The execution of this methods is in charge of the (human and non-human) resources that participate in the workflow. On the other hand, composite methods *decompose* into subtasks and specify the control structure over the subtasks. The *operational description* of these methods is the meeting point with control structure of workflows. Current implementations of the UPML framework represent the control-flow in a program like way (Motta 1998) and are not well suited for being reused. Our framework deals the operational description of the method as a new component that defines the control of the execution coordination of the method subtasks. This solution facilitates the reuse of both the *methods* and the *control models*:
  - Several control structures can be associated (through a bridge) with the same method, which enriches the selection of the most suitable method configuration for solving a task. A business level criteria may define which is the best control structure for a given method. This criteria may be based on organization practices, logistics, fault tolerance, and so forth.
  - The same control structure can be reused for different methods because it does not depend on the method decomposition. In our framework, the subtasks are mapped by a bridge with the control structure. Therefore both the method (and its task decomposition) and the control structure (on the tasks) can be defined and reused independently.
- *Domain*. A *domain model* describes the knowledge of a domain of discourse. The domain model introduces domain knowledge, merely the formulas that are then used by problem-solving methods and tasks.
- *Ontology*. An ontology defines a terminology and its properties, used by resources, control structures, tasks, problem-solving methods, and domain models. The core of an ontology specification is its signature definition, which defines signature elements that hold terms. The ontology also provides the axioms that characterize logical properties of the signature elements.

Control Model

The *control model* deals with the definition of the control-flow specifying the activities coordination and the conditions that they must verify to enable their execution. A control model is a knowledge component of the proposed framework that captures a process-oriented perspective of the workflow. This model will provide the execution level semantics (Sheth & Gomadam 2007; Sivashanmugam *et al.* 2003) of the control structure and the reusable structures that methods will adapt to define their operational description.

Process modeling techniques are wide and heterogeneous: process algebra's, Petri nets, and vendor specific diagrams are the most representative solutions. At present there is no standard language for workflow specification, but in our opinion a solid theoretical foundation with a graphical semantics is needed to make the definition of processes easier.

In this framework, we used *high-level Petri nets* (ISO/IEC 15909-1 2002) to design processes structure, because *(i)* they are a formalism with a graphical representation and *(ii)* they provide the explicit representation of the states and events of processes. In order to incorporate the process structure as a knowledge component, we have developed a *high-level Petri net ontology* (Vidal, Lama, & Bugarín 2006a) which captures both the static elements and dynamic behavior of this kind of nets in a declarative way. The ontology describes high-level Petri nets as graphs which is a common way to describe these kind of nets. Based on this representation, this component provides the properties that allow the definition of these graphs. For example, they define:

- The *nodes* which capture the vertices of the graph. High-level Petri nets are bipartite directed graphs and thus are defined by two disjoint sets of vertices called places and transitions. As it is depicted in Figure 3, places are graphically represented as circles while transitions as rectangles.
- The *arcs* capture the directed edges that connect a source node to a target node. An arc connects places with transitions and transitions with places but never nodes of the same type.
- The *signature* define the set of sorts and operators a high-level Petri net can use. These sorts and operators will be used to define the algebraic specification for annotating the places, transitions and arcs of the net. As it is depicted in Figure 3, a place is annotated with a sort (concept) at the top of the circle (e.g. *Case*), a transition is annotated with a condition in brackets at the bottom of the rectangle which define its enabling condition (e.g. *[true]*), and an arc is annotated with a term that can be a variable (e.g. *x*) or an operator application (e.g. *F(x)*).

In our framework, a control structure, as any other knowledge component, uses ontologies to define its properties. In this case, the ontologies will be used to define the signature of the net, that is, the sorts and operators used to annotate the net.

Resource Model

Although the main attention of workflow researchers has focused on the process perspective, i.e. the control-flow, the resource perspective, i.e. the people and machines actually doing the work, is essential for successful implementation of workflow technology.

Our framework addresses this issue through the *resource model* (see Figure 2) which structure is based on the TOVE ontology (Fox & Gruninger 1998). We created this model as a special class of domain model that must be explicitly defined because it plays an important role both in the definition and execution of workflows. In this sense, a resource model cannot be designed like a classic domain model since *(i)* each resource of the model needs to define a binary relation with the domain model itself in order to define the knowledge it manage and *(ii)* each resource also needs to define the methods it can execute.

The *resource model* extends therefore the domain model with the properties of the TOVE ontology. In this context, a resource is a member of an organization that is capable of doing work. A resource has a specific position in that organization with specific privileges associated with it. He may also be a member of one or more organizational units which are permanent groups of resources within the organization, e.g. product department, development unit. A resource may also have one or more associated roles. Roles serve as another grouping mechanisms for human resources with similar job roles or responsibility levels, e.g. managers, technical (Russell *et al.* 2004).

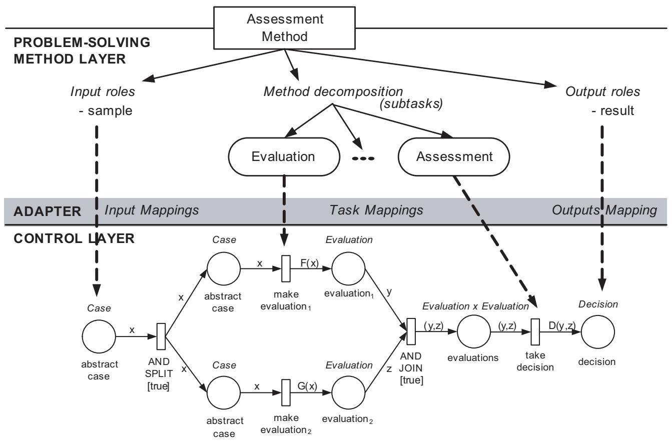

Figure 3: Relation between the problem-solving method and the Petri net that defines the control structure of the workflow. The mappings between the two layers are defined with dotted arcs.

Our framework defines two types of resources: human and non-human. We restrict non-human resources to external services which call will be in charge of the workflow system. In the case of human resources, the system does not take the initiative and the resources (users) are in charge of retrieving their work list, performing their work and notifying their results. Thus, tasks assigned to non-human resources are called automatic tasks. In any case, resources will be in charge of the execution of non-composite methods.

Bridges

The reuse of the knowledge components is achieved through the adapters such as they are defined in the UPML framework (Fensel *et al.* 2003). The framework provides two types of *adapters* to define binary relationships between knowledge components: bridges and refiners. Bridges explicitly model the relationships between two distinguished knowledge components, while refiners are used to constraint the definition of a knowledge component. Besides the bridges that are defined in UPML for describing problem-solving methods, our framework adds two bridges:

- *Method-Control bridge* relates a composite method with a high-level Petri net. As it is depicted in Figure 3, this bridge maps:
  - The inputs and outputs of the method with places of the

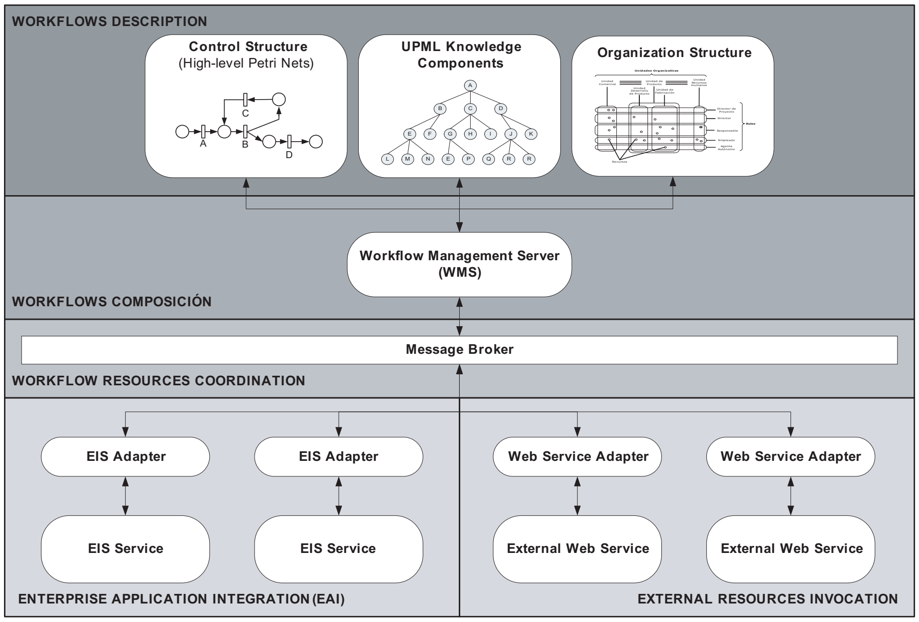

Figure 4: Infrastructure for the execution of knowledge-enriched workflows

Petri net (e.g. the *sample* input with the *abstract case* place).

- The subtasks of the composite methods with the transitions of the Petri net (e.g. the *evaluation* task with the *make evaluation*$_1$ place).
- The signature used in the definition of the method with those of the Petri net, that is, it provides a different interpretation of the annotations, sorts and operations of the Petri net.

- *Resource-Domain Model bridge*. Individual resources may also possess capabilities that further clarify its suitability for executing various kinds of tasks. These may include qualifications and skills as well as other attributes such as specific responsibilities held or previous work experience. They features could be of interest when allocating tasks (work items).

Finally, we must mention the behavior of the *Domain-Method bridge* when this bridge relates a resource with a method. In this case, it relates the roles, units and participants of the resource model with a method. When a method is non-composite, then this association implies that the resource will perform the method, e.g. a web service. Resources cannot be associated to composite methods since these methods are decomposed in subtasks and will be solved by another methods.

Workflows Execution

Taking the previously described framework into account, Figure 4 depicts the infrastructure for a service-oriented execution of knowledge-enriched workflows. It defines a four layers model which captures the components that are involved in the execution:

- The first layer of the infrastructure provides the access to the *description* of the workflows that must be executed. This layer creates this description from its knowledge components, that is, from the control structures, organization structures, problem-solving methods, tasks, and domain models. Following the philosophy of UPML, these components are glued through a set of adapters which define the way these knowledge components can be related and the conditions under which they can be combined. This feature makes the reuse of workflows easier since it only implies the redefinition of the adapters.
- The second layer handles the *execution* and *composition* of the workflows. This layer uses the first layer in order to obtain the workflow description and thus each one of the tasks that must be executed. This layer performs the following operations:
  - *Execution*. The execution of a task implies that the most suitable method and resources must be selected in order to drive its execution. When the selected method is non-composite, this entails the selection of the most suitable *(i) resource-method* adapters that relate the task that must be performed with the resources that have permissions to perform it and *(ii) resource-domain* adapters that match the knowledge required by the tasks with those provided by the resource. From this information, the scheduler of the workflow engine assigns the work to the most suitable resource (the intersection between the two adapters). When the work is assigned to a software resource then the workflow engine *(i)* searches the service description in its repository (which acts as a service registry) and *(ii)* call this service through the third layer of the infrastructure.

  Figure 5 depicts the knowledge components that participate in a typical execution of a workflow. If we obviate the resources, a workflow is described as a task that is solved by a problem-solving method which operational description is defined as a Petri net-based process structure. The first layer of the Figure 5 represents this structure where each one of the transitions depicted in the net represents a task to be performed. Therefore, this example describes the structure of a composite problem-solving method composed of four tasks (second layer): *abstract*, *evaluate case*, *revise case*, and *create order*. In the third layer of the Figure 5 a problem-solving method is assigned to each task. When the method that solves the task is non-composite, such as the method that solves the *abstract* task, then the tree structure defined by the execution has reached a leaf. However, when method is composite, such as the *evaluate case* method that solves the *CAD assessment* task, then a new non-leaf node is defined. This new node will define another execution structure similar to those depicted in Figure 3, that is, the method will define a new tasks decomposition structured according to a Petri net. In the *assessment* method scenario, the method is composed of two tasks (*Evaluation* and *Assessment*) that are mapped with two transitions of the Petri net that define its control structure.

  – *Composition*. When the method that solves a task is composite then it will be composed by a set of tasks controlled by another Petri net structure. The composition between the different Petri nets implies the definition of a hierarchical net (Gomes & Barro 2005). The transition that represents the task solved by the composite problem-solving method is substituted by the Petri net that defines its control structure.

    The *assessment* method depicted in Figure 5 is subject of such composition although it is not detailed in the figure. In this case, the *evaluate case* transition (task) of the upper Petri net must be linked with the lower Petri net that substitutes the *assessment* method. That is, the input and output places of the *evaluate case* transition are fused with those of the substitute net. As result of this fusion, the behavior of the *evaluate case* transition is assumed by the lower Petri net.

- The third layer of the infrastructure facilitates the *coordination* of the resources that participate in the workflow. This layer is defined by a *message broker* which establishes the logic of the messages exchange. Through this solution the heterogeneity of the resources that participate in the workflows is hidden by the message broker. This architecture also uses the publication/subscription interaction model.
- Finally, the fourth layer depicted in Figure 4 *integrates* the systems that will execute the non-composite methods, such as the Enterprise Information Systems (EIS) or Web services published by external providers. This integration is performed through adapters which will participate in the execution like any other resource. This layer enables the access to all the systems with the same programming model and data formats.

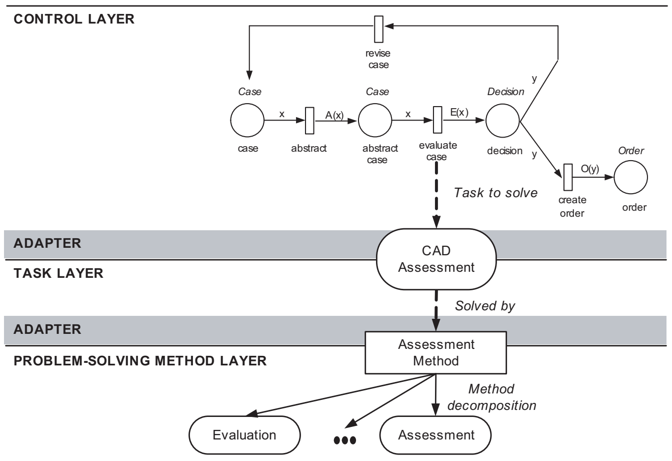

Figure 5: Example of execution of a knowledge-enriched workflows

Conclusions and Future work

This work describes a new knowledge-based framework for workflow specification. The proposed framework extends the UPML framework for problem-solving method description with (i) a high-level Petri net ontology for describing the operational description of composite methods and (ii) an organization ontology for describing the organizational structure. With these extensions the definition of a workflow can be viewed as the specification of a problem-solving method that includes the process structure (through a Petri net) as the operational description and, when it is necessary, the external agents that execute a non-composite method.

Our framework is related with current proposals for defining an ontology-based infrastructure to develop, discover and compose semantic web services (de Bruijn *et al.* 2005; Battle *et al.* 2005; Martin *et al.* 2004). Particularly, the WSMO approach (de Bruijn *et al.* 2005) shares with our framework that is also based on the UPML specification, and therefore it inherits from UPML the concept of ontologies, tasks (aka *goals*), and adapters (aka *mediators*), which enable the connection between all the components of the framework. The main difference between our approach and WSMO is that WSMO does not introduce the concept for workflow in its specification: it uses abstract state machines to define the structure of the choreography and orchestration of semantic web services. Our approach, however, considers as an assumption the use of workflows, and particularly of Petri nets, because they are accepted in the industry and academic domains as a paradigm for process modeling.

From this perspective, the main advantage of the proposed framework is due to its architecture which facilitates the definition of workflows through a set of knowledge components. Since each one of the components is defined independently from the others, this framework facilitates the reuse and composition of workflows: through the bridges different parts of several workflows (tasks, process structure, organization description, etc.) could be used to compose a new workflow.

Based on this framework a service-oriented architecture for the definition and execution of workflows has been developed (Vidal, Lama, & Bugarín 2007). This architecture is currently being applied in the domains of (1) furniture industry where it is being used to define the business processes associated to the creation of designs and assembly of the pieces of a given furniture (Vidal, Lama, & Bugarín 2006b), and (2) eLearning for the execution of learning designs which are modeled through workflows with teachers and students as the participants that execute the learning tasks.

Acknowledgments

Authors wish to thank the Xunta de Galicia and the Ministerio de Educacin y Ciencia for their financial support under the projects PGIDIT06SIN20601PR and TSI2007-65677C02-01.

References

Andrews, T.; Curbera, F.; Dholakia, H.; Goland, Y.; Klein, J.; Leymann, F.; Liu, K.; Roller, D.; Smith, D.; Thatte, S.; Trickovic, I.; and Weerawarana, S. 2003. *Business Process Execution Language for Web Services Version 1.1*. IBM. Available at http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/.

Battle, S.; Bernstein, A.; Boley, H.; Grosof, B.; Gruninger, M.; Hull, R.; Kifer, M.; Martin, D.; McIlraith, S.; McGuinness, D.; Su, J.; and Tabet, S. 2005. *Semantic Web Services Framework (SWSF) Overview*. World Wide Web Consortium (W3C). Available at http://www.w3.org/Submission/SWSF/.

Cabral, L.; Domingue, J.; Galizia, S.; Gugliotta, A.; Tanasescu, V.; Pedrinaci, C.; and Norton, B. 2006. IRS-III: A broker for semantic web services based applications. In *International Semantic Web Conference*, 201–214.

de Bruijn, J.; Lara, R.; Arroyo, S.; Gomez, J. M.; Sung-Kook, H.; and Fensel, D. 2005. A Unified Semantic Web Services Architecture based on WSMF and UPML. *International Journal on Web Engineering Technology* 2(2):148–180.

Fensel, D.; Motta, E.; Benjamins, V. R.; Crubezy, M.; Decker, S.; Gaspari, M.; Groenboom, R.; Grosso, W.; Musen, M.; Plaza, E.; Schreiber, G.; Studer, R.; and Wielinga, B. 2003. The Unified Problem-solving Method Development Language UPML. *Knowledge and Information Systems* 5(1):83–131.

Fox, M., and Gruninger, M. 1998. Enterprise Modelling. *AI Magazine* 109–121.

Gomes, L., and Barro, J. P. 2005. Structuring and Composability Issues in Petri Nets Modeling. *IEEE Transactions on Industrial Informatics* 1(2):112–123.

ISO/IEC 15909-1. 2002. High-Level Petri Nets - Concepts, Definitions and Graphical Notation.

Martin, D.; Burstein, M.; Hobbs, J.; Lassila, O.; McDermott, D.; McIlraith, S.; Narayanan, S.; Paolucci, M.; Parsia, B.; Payne, T.; Sirin, E.; Srinivasan, N.; and Sycara, K. 2004. *OWL-S: Semantic Markup for Web Services*. World Wide Web Consortium (W3C). Available at http://www.w3.org/Submission/OWL-S/.

Motta, E. 1998. An Overview of the OCML Modelling Language. In *Proceedings of KEML'98: 8th Workshop on Knowledge Engineering Methods and Languages*.

Newell, A. 1982. The knowledge level. *Artificial Intelligence* 8(1):87–127.

Object Management Group (OMG). 2006. *Business Process Modeling Notation (BPMN) Specification Final Adopted Specification*. Available at http://www.bpmn.org/Documents/.

Russell, N.; ter Hofstede, A.; Edmond, D.; and van der Aalst, W. 2004. Workflow Resource Patterns. BETA Working Paper Series WP 127, Eindhoven University of Technology.

Sheth, A. P., and Gomadam, K. 2007. The 4 x 4 semantic model: Exploiting data, functional, non-functional and execution semantics across business process, workflow, partner services an middleware serices tiers. In *9th International Conference on Enterprise Information Systems (ICEIS 2007)*, 1–4.

Sivashanmugam, K.; Verma, K.; A., S.; and Miller, J. 2003. Adding semantics to web services standards. In *1st International Conference on Web Services (ICWS'03)*.

Vidal, J. C.; Lama, M.; and Bugarín, A. 2006a. A High-level Petri Net Ontology Compatible with PNML. *Petri Net Newsletter* 71:11–23.

Vidal, J. C.; Lama, M.; and Bugarín, A. 2006b. *Integrated Intelligent Systems for Engineering Design*. Frontiers in Artificial Intelligence and Applications. IOS Press. chapter Integrated Knowledge-based System for Product Design in Furniture Estimate, 345–361.

Vidal, J. C.; Lama, M.; and Bugarín, A. 2007. Service-oriented architecture for knowledge-enriched workflows modelling and execution. In Abramowicz, W., and Maciaszek, L., eds., *Business Process and Services Computing*, Lecture Notes in Informatics, 69–77.

Workflow Management Coalition (WfMC). 2005. *Process Definition Interface - XML Process Definition Language*. Available at http://www.wfmc.org/standards/docs/.