An Adaptive Process Management System (APMS)

Christopher Menzel and Perakath Benjamin
Knowledge Based Systems, Inc., U.S.A.
Em: cmenzel@kbsi.com

Keywords Knowledge sharing, integration architectures, enterprise integration, process management

Abstract The increasing complexity of manufacturing systems coupled with the emergence of truly global manufacturing enterprises requires the use of novel approaches to facilitate enterprise integration. Specific challenges include (i) the lack of theoretical foundations for enterprise integration and (ii) the absence of mechanisms for knowledge sharing and semantic interoperability. This paper presents a process-centered approach to address these problems. Specifically, the paper describes the theoretical foundations and concept of operation of an integrated architecture for process management, the Adaptive Process Management System (APMS). APMS facilitates information-integrated management of the Product Realization Process (PRP) using a life-cycle perspective.

1 INTRODUCTION

The rapid increase in the speed and sophistication of computers in recent years have led to a corresponding increase in the number computer-based enterprise modelling applications. Among the most important of these are applications that support the creation of *dynamic* models: models that represent information about how things within a portion of an enterprise change, or ought to change, over time – in a word, information about enterprise *processes*. Such models are especially critical to a business' ability to respond rapidly and creatively to change.

Cross-functional processes are the focus of many different business functions. Consequently, there are a wide variety of process-oriented applications for the creation and maintenance of enterprise process models that support those functions: simulation, project management, process and production planning, scheduling, workflow management, and so on. Thus,

J. P. T. Mo et al. (eds.), *Global Engineering, Manufacturing and Enterprise Networks*

to alter its enterprise processes, a company must alter the models it maintains across these applications. But there is a serious problem: like the builders of the Tower of Babel after the confounding of tongues, these applications cannot talk to each other. If, however, the applications that harbor an enterprise's models cannot communicate, then there is no reliable way for the implications of changes in one model to be propagated appropriately to all related models; one must rely upon human agents managing the processes to communicate and to do so quickly, knowledgeably, and accurately – attributes not often associated with human communication within a modern enterprise.

The idea of an interlingua is not new – such a language for exchange of information between (typically static) knowledge bases was in fact developed as a part of the DARPA-sponsored Knowledge Sharing Effort (KSE) [1]. More recently, however, two projects have turned their attention to the development of an interlingua that is tailored specifically to address the problem of integration among dynamic modelling applications: the Process Interchange Format (PIF) project [2] and the NIST Process Specification Language (PSL) Project [http://www.nist.gov/psl].

2 FRAMEWORK FOR PROCESS KNOWLEDGE SHARING

2.1 The Process Specification Language (PSL)

The PSL consists of several components, or *theories*, based on first-order logic. The fundamental theory is known as *PSL core*. The idea behind PSL core is to provide a theory that characterizes a basic ontology of dynamic information consisting of four basic types of entity: activities, activity occurrences, objects, and timepoints. The essential characteristics of these entities are captured in a series of *axioms* expressed in a precise logical language. Examples of such principles are that timepoints are linearly ordered (by the *temporal precedence* – a.k.a. *before* – relation), and that the ending timepoint of an activity occurrence cannot precede its beginning point. Extensions of PSL capture the properties of activity resources, subactivities, and complex activities.

2.2 Process Specifications for Knowledge Sharing

When one describes the general structure of a process—whether graphically, or by means of a structured text file or a well-defined representation language—one is giving a *process description*. A primary function of PSL is to provide a single framework that is capable of expressing the content of any process description in any reasonable

application representation language. Any two applications that have translators into and out of PSL will then be able, as far as possible, to share information despite the fact that neither "speaks" the other's "native tongue." The Adaptive Process Management System (APMS) is an integrated process-oriented architecture that implements the PSL framework. Section 3 outlines an example application of the APMS architecture.

The current implementation of PSL approaches the specification of processes by identifying processes with a certain class of "complex" activities—activities whose instances can vary greatly in form and which are specified by means of complex logical descriptions. A typical example of such an activity would be one that is specified by an if-then-else conditional: "If condition φ holds, then activity A occurs, otherwise B occurs." An occurrence of the complex activity so described could consist in either an occurrence of A or an occurrence of B depending on whether or not the precondition holds. A process specification in a given application language would be translated into set of logically complex PSL statements that jointly define a complex activity. Complex activities and their description, however, are indeed rather complex, and the extension that introduces them into PSL involves quite a lot of heavy theoretical machinery. For many purposes, when one needs to capture the content of highly detailed and fine-grained process descriptions, this additional power is necessary. However, as often as not, a simpler framework that builds only upon PSL core will suffice. More exactly, on this approach, there are no complex activities; i.e., there are no processes, *per se*. Rather, there are only PSL-based process specifications; i.e., certain syntactic constructions that can be given a semantics in terms of the semantics of PSL core. To define a PSL process specification, we must first define the notion of an *activity role specification*. To define this idea precisely, we need a complete characterization of the notion of a PSL language, which would take us rather too far afield.

A process specification, then, is simply a set P of activity specifications that satisfies two conditions: (i) no two activity role declarations in P can have the same numerical identifier (i.e., the same value in their :id fields), and (ii) every identifier in the :successors field of an activity role declaration in P must be the numerical identifier (i.e., the value of the :id field) of some activity role declaration in P. Condition (i) is the formal mechanism that enables us to represent activity roles (as this allows us to specify the same activity in the name fields of distinct activity roles. Condition (ii) ensures that there are no "dangling" successor relations in a process—i.e., cases where an activity role is declared to have a successor in the structure of the specified process by specifying an activity ID in the "successors" field, but

where there is, in fact, no corresponding activity role with that ID in the specification.

3 APMS APPLICATION SCENARIO

We will now outline an application of the APMS integration architecture for knowledge sharing. We use a simple example to illustrate how APMS facilitates knowledge sharing through the use of the PSL. The knowledge sharing occurs in this example through *reuse*: the ability to import the information from a model M1 in one application A1 into a model M2 from another application A2 so that the information garnered in the first needn't be reconstructed from scratch in the second. The application scenario is as follows (Figure 2). Mr. Dawn, ACME Inc.'s production manager, would like to validate and implement a new Widget manufacturing process. He performs the following activities: (i) develop a process plan for making the Widget, (ii) analyze the plan to validate process performance, (iii) formulate an implementation schedule and dispatch resources, and (iv) implement the process and monitor performance. PSL is used in this scenario to enable the transfer of information between (a) a process planning model, (b) a simulation model and (c) a scheduling model.

3.1 The Process Plan

The process plan is developed using the IDEF3 process modelling language (http://www.idef.com).

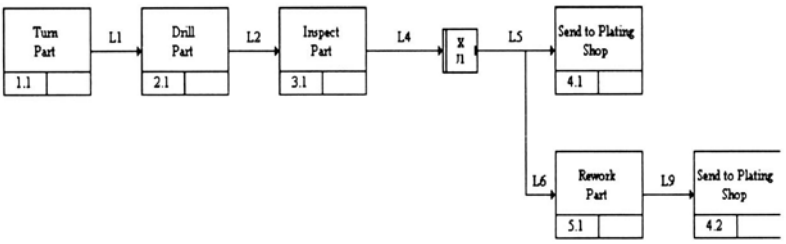

Figure 1: Portion of the Widget-Making Process Plan

This graphical aspect of the model is self-explanatory (Figure). The planned Widget-making process involves turning a part and then drilling it, followed by an inspection. If the part passes the inspection, it is sent immediately to the plating shop. If not, it is sent for rework before being sent to the plating shop.

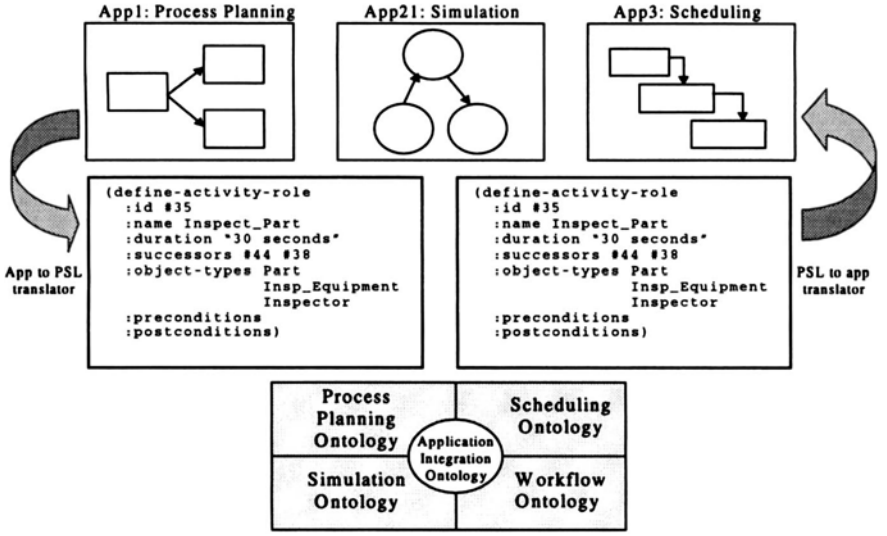

Figure 2: APMS Facilitates Integration through Knowledge Sharing

In addition to the graphical information, an IDEF3 model also enables one to store information about the resources needed for each activity in the process as well as its duration. The type of information that can be stored is shown in Table 1.

Table 1: Activity Specification

Activity Name	Resources	Processing Time
Turn Part	Lathe, Turning Operator, Turning Tool	20 seconds
Drill Part	Drilling Machine, Drilling Operator, Drilling Tool	15 seconds
Inspect	Inspection Equipment, Inspector	30 seconds
Send to Plating Shop	Transporter	5 seconds
Rework Part	Rework Tool, Rework Operator	25 seconds

3.2 The Schedule

The process plan is now used as the basis for generating an implementation schedule. The implementation schedule includes (i) a calendar of tasks, (ii) start and finish dates including an identified critical path, and (iii) a resource utilization chart. The schedule is used as the basis for dispatching work to the widget manufacturing shop floor resources (people, machines, and tools). The schedule is also integrated with the shop floor work status monitoring systems. The schedule is updated regularly

based on (a) changes in execution status and (b) changes in the manufacturing plan requirements (Figure 3).

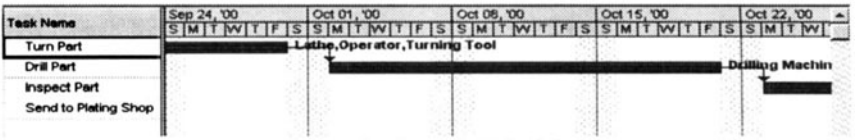

Figure 3: Example Schedule

Additional information is required by the schedule model over and above the information captured in the IDEF3 process planning model: the *scheduled plan instantiation paths*. In this example, the IDEF3 "X" Junction involved a decision (based on the quality of the part) that would result in the selection of one out of two possible instantiation paths. Recall that the simulation model required the specification of the decision rule for the decision. Scheduling requires the representation of a detailed schedule (tasks and resources represented on a calendar) for each instantiation path. Moreover, some widely used scheduling tools (such as MSProject) do not allow for the representation of multiple, mutually exclusive paths in a schedule.

4 HOW PSL FACILITATES KNOWLEDGE SHARING

In our scenario, a variety of application-specific tools from difference vendors are used in the development and implementation of a process. This is the typical case. It is also typical that there is no way to share the information in one application with another. Thus, if the above scenario were played out along the usual lines, Mr Dawn would have to recreate information in each successive model that is present in the preceding models. Such recreation is wasteful of time and resources, and is also prone to error and inconsistency. It would be preferable if information, once recorded in a given model, can be *reused* in successive models, thus saving the time and resources that are lost in information recreation and avoiding the high potential for error and inconsistency.

For reasons discussed in Section 1, the development of pairwise translators between applications is not feasible. PSL provides the representational interlingua between applications that brings the benefits of knowledge sharing without the liabilities of pairwise translators. How is this accomplished? The interlingua architecture is predicated upon the existence of application representation languages. However, many process-oriented modelling applications, such as IDEF3 and MS Project, are graphically based. How can PSL then be used?

The answer to this question is that most all modelling applications have the ability to translate the contents of a model into some sort of text-based file format. Such formats can be thought of as rudimentary representation languages. Consider the IDEF3 representation of a process plan for making widgets, generated by means of the PROSIM process modelling tool. To represent the information in the model in a non-graphical form, PROSIM can translate its graphical models into a well-defined textual representation language. Thus the above model—together with information about participating object types and instantiation constraints not pictured in Figure —translates into a structured text.

In essence, this process description is stated in a perfectly respectable representation language, albeit one that is tailored toward the eccentricities of IDEF3. There is, in fact, no explicit mathematical semantics for this language (or for IDEF3 generally), but one could clearly be given. It is also reasonably clear how this language can be translated directly into a PSL process specification. For instance, from the information for `Inspect_Part`:

```
Process 34
  Name: Inspect Part
  ID: #35
  Duration: 30 seconds
    Object Occurrence List
      Object Occurrence 54 2 (Part)
      Object Occurrence 106 1 (Inspection Equipment)
      Object Occurrence 107 1 (Inspector)
    End Object Occurrence List
End  Process 34
```

It is possible to extract the contents of the :name, :id, Duration:, and :object-types fields of a PSL activity declaration. And, while links do not correspond to anything in a PSL process specification but instead from information from the IDEF3 links, it is possible to extract information for the :successors field. We would thus have the following activity role declaration:

```
(define-activity-role
  :id #35
  :name Inspect_Part
  :duration "30 seconds"
  :successors #44 #38
  :object-types Part Inspection_Equipment Inspector
  :preconditions
  :postconditions)
```

The logic of the XOR junction would be expressed as preconditions on the Send to Plating Shop activity with ID #38 and on the Rework Part activity. For example, in the case of the former we would have

```
(define-activity-role
  :id #38
  :name Send_To_Plating_Shop
  :duration "5 seconds"
  :successors
  :object-types Part Transporter
  :preconditions (PassedInspection (the ?x (Part ?x)))
  :postconditions)
```

whereas the `:preconditions` field for `Rework Part` would be the negation `(not (PassedInspection (the ?x (Part ?x))))`.

As it happens, PROSIM uses a logical language to express constraints that is very similar to PSL, but in general this will not be the case. Hence, for applications that include a logical language, part of the translation process will have to include a "compiler" for rendering statements in the application language in PSL.

Microsoft has developed a text-based form for the information in a Project schedule known as MPX. Again, as with PROSIM, this can be viewed as a tailored representation language. Thus, reuse of the PROSIM process plan can be achieved via translation from PSL process specifications into MPX format. This, in fact, can be easily accomplished. Adding certain pieces of scheduling boilerplate (currency formats, work hours, etc), the PSL representation of the PROSIM schedule can be translated into two MPX files corresponding to the two possible instantiations that can arise from the XOR of the process model.

5 APMS BENEFITS

5.1 APMS Facilitates Semantic Integration

We conclude from the previous section that the APMS architecture solves one of most critical problems for semantic integration: *conflicts in terminology*. These conflicts can be of two sorts. First, the same term can be used to denote different concepts (the *ambiguity* problem). Second, different terms can be used to denote the same concept (the *synonymy* problem). The ambiguity problem is eliminated because different concepts are represented differently in PSL. Since application representation languages with ambiguous terms do not directly "speak" to each other, ambiguity will be filtered out via intermediate PSL translations. Similarly, the synonymy problem will be avoided because the different terms will be mapped to the same representation in PSL, and hence translators will ensure that synonymous terms in different applications are properly mapped to one another.

5.2 APMS Provides a Robust Integration Ontology

There is a further important element to the PSL integration architecture that deserves mention. PSL is part of a growing, global effort to develop large ontologies of fundamental clusters of concepts for next generation knowledge management applications. Part of this effort consists in the development of a number of general process-oriented ontologies targeted by different classes of process-oriented applications. Thus, in addition to PSL's general ontology of processes, there are also more specialized ontologies for process planning, simulation, scheduling, and workflow that are being developed. However, these specialized ontologies alone are still not enough for robust knowledge sharing. In terms of a logical connection between a process plan and a corresponding schedule, is not something present in an ontology of either process planning or scheduling. Rather, it is what's known as an *integration axiom*—it relates concepts across an ontological divide. A collection of such axioms for a given set of ontologies is known as an *integration ontology*. Integration ontologies are critical for ensuring that the maximum possible information in a model of one sort is available for sharing and reuse in a model of another sort. Consequently, the use of integration ontologies forms a critical part of the overall PSL-based architecture for knowledge sharing among process-oriented applications.

6 CONCLUSION

This paper described a process-oriented theoretical framework and architecture (APMS) for facilitating enterprise integration through knowledge sharing. The role of the PSL in facilitating semantic interoperability between process-oriented applications was described. An example application scenario was used to illustrate the role of PSL for knowledge sharing. APMS provides important enabling technology for the management of information-integrated manufacturing enterprises.

7 REFERENCES

[1] Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator T., and Swartout, W., (1991), "Enabling Technology for Knowledge Sharing," *AI Magazine* 12(3) 36-56.

[2] Lee, J., Grunninger, M., Jin, Y., Malone, T., Tate, A., Yost, G. (1996). "The Process Interchange Format (PIF) and Framework Version 1.1," MIT Center for Coordination Science, Working Paper #194.