Submitted to the Scientific Committee of the 12th European Symposium of Computer-Aided Process Engineering, The Hague 2002

Representing recipes and procedures with PSL

Rafael Batres, Atsushi Aoyama and Yuji Naka

Tokyo Institute of Technology, R1 Bldg., Midori-ku, Nagatsuta 4259, Yokohama 226-8503, Japan

This paper explains how a process ontology provides the common syntax and semantics to represent batch recipes and operating procedures. Ontologies define unambiguous descriptions of concepts in a particular domain, playing an important role in the integration of software and people. Moreover, ontologies constitute the basis of a new generation of the World Wide Web known as Semantic Web, where information can be shared and exchanged in a way that all the involved parties share the same meaning of the terms.

1. INTRODUCTION

Very often, documents are written in a way that fail to describe a complete operating procedure or recipe. For example, recipes developed at the process design stage may include only information about raw materials and their quantities together with a brief description of the process. The management of these documents becomes error-prone and time consuming in situations where recipes are needed for different plants with similar products.

Similarly, operating procedures and procedural control descriptions are often ambiguous which results in operation delays and hazardous situations. Moreover, equipment aspects are mixed with operational and process descriptions contributing to inconsistencies during changes in the plant that limit the flexibility of using a recipe for different plants or groups of equipment. The terminology and recipe structure described in batch control S88 standard has reported benefits in overcoming this situation [1]. Despite that the S88 committee has proposed a data structure for representing recipes, still part of the process and control information in the recipe is left to textual descriptions in natural language. Furthermore, integrators rely on personal interpretations on the standard.

Ontologies that define a common agreed view of the product, process, and plant life-cycles play a very important role in the integration of engineering software tools [2]. Moreover, ontologies constitute the basis of a new generation of the World Wide Web known as Semantic Web, where software agents and people can share and exchange data in a way that all the involved parties share the same meaning of the terms describing the data without implicit assumptions [3].

The Process Specification Language (PSL) project at the National Institute of Standards and Technology (NIST) is developing a neutral, standard interchange language for the specification of processes (such as production plans) to integrate multiple applications throughout the life cycle [4]. An ontology with formal definitions represented in KIF (Knowledge Interchange Format) are developed so that information can be exchanged without relying on hidden assumptions or implicit knowledge. The ontology supports temporal constraints, aggregation and decomposition of activities. In this paper, we describe how an extended PSL ontology could be used to describe recipes and procedures in an unambiguous way. To define references to material and equipment links are made to other ontologies that specify such aspects.

2. PSL ONTOLOGY

An ontology is a formal and explicit representation of unambiguous definitions of objects, concepts and the relationships that hold among them. The role of an ontology is to enable consistent interpretation of terms that participate during information sharing and exchange.

From business plans to plant operations, managers, engineers and plant personnel deal with plans or procedures to achieve objectives constrained to a number of factors. Plans can be seen as activities that are made up of other activities that can be broken down further in a hierarchical fashion. PSL is an interchange ontology for the exchange of activity information among pieces of software such as scheduling applications, workflow, and project management software. The semantics of the concepts and their relationships is based on the Knowledge Interchange Format (KIF) specification [5].

The PSL can be organized in two parts: PSL Core and extensions. PSL Core contains the basic elements of PSL. The main concepts are *activity*, *activity_occurrence*, *timepoint*, and *object*. The key concept *activity* is the template for a specific event or action. Occurrence is the instance of the activity that takes place over a time interval. Timepoint is used to reason about the duration of an activity. Object is a concrete or abstract thing that can participate in an activity. Examples of objects are plant operators, plant equipment, though abstract objects. In PSL, objects can be created and get consumed (such as a raw material) at certain points in time.

Extensions of the PSL core define aspects such as the relation of activity occurrences and the world, the state of the activity, duration and ordering, preconditions and effects, etc. The interaction of the world is defined around the concept of *fluent*. A fluent is a property of the world that can change as a result of an activity occurring. A fluent could be a valve state, process variables, material properties, information or money. For operation and control, in the lowest level of an activity hierarchy, an activity occurrence results in changes to the state of the actionable equipment (such as opening a valve). Information about the initiating agent or performer of the activity is stored by the activity for coordination purposes.

An activity can be decomposed into *subactivities*. In this case an activity precondition of an activity initiates not one but a series of subactivities. For example, if an activity precondition is activated to change the state of a multi-stage compressor from off to on, then a series of operations will be initiated to modify the state of each stage from off to on.

To reason about the state of an activity, PSL introduces the concept of *interval-activity*. An interval-activity is an activity that has an initial subactivity that occurs at the beginning and a terminal subactivity that occurs at the end of each occurrence of the interval-activity. The initial activity establishes a particular property called activity-fluent that describes the occurrence of the activity. The terminal activity falsifies that property. S88 commands and states for procedural control can be represented with this concept. S88 suggests the running, holding, restarting pausing, stopping, aborting, idle, complete, held, paused, stopped, and aborted states along with the start, hold, restart, pause, resume, stop, abort and reset commands. For example, among the combination of commands that initiate and terminate the running state, the following KIF expression holds for running when initiated with the start command and terminated with the stop command:

```
(=>  (running ?a)
     (and
          (start ?a1)
          (stop ?a2)
          (initiate ?b)
          (terminate ?c)
          (= ?a1 ?b)
          (= ?a2 ?c)))
```

Conditional execution of activities is possible with the possibly-changes command. The *possibly-changes* concept is used to capture the effect of an activity-occurrence when that effect is conditional on some property of the world. An activity-occurrence possibly changes a fluent f2 given a fluent f1, if it changes f2 only when f1 holds before the activity-occurrence. A use of this concept is in the representation of preconditions of an operation such as “if pressure at PIC-06 > 75 Kgf/cm2G and temperature at TY-28 > 175C then start operation1”.

Another useful extension is the theories of resources that presents potential uses for the allocation of equipment to the various operations during batch scheduling. Relations exist that describe exclusive or shared uses of resources, an important distinction in S88.

3. PROCESS ENGINEERING ONTOLOGIES

The approach followed to develop the process engineering ontologies utilizes the Multi-dimensional Formalism (MDF) as its basis to organize ontologies for representing the knowledge associated to product, processes and plants.

MDF defines a modeling scheme that organizes information, activities and tools into structural, behavioral, and operational dimensions or perspectives [6]. Figure 1 shows a simplified view of the organization of the ontologies.

```
🗁Plant ontology
     🗁 Plant-Behavioral-Dimension
     🗁 Plant-Management-And-Operation-Dimension
     🗁 Plant-Structural-Dimension
          🗁 Plant-System-Ontology
          🗁 Facility-Ontology
🗁 Process ontology
     🗀 Process-Behavioral-Dimension
     🗀 Process-Management-And-Operation-Dimension
     🗀 Process-Structural-Dimension
🗀 Product ontology
🗀 Management and Operation Generic Ontology
🗀 Topology and Mereology Ontology
🗀 Generic Behavior Ontology (Metamodeling
```

Figure 1. Organization of the ontologies

The management and operation generic ontology specifies classes, axioms and relations that describe objects associated to a variety of management, control, and operational tasks. These tasks run from management, planning and scheduling, and plant-wide control through local advanced and regulatory control. The management and operation generic ontology specifies concepts that are common to ontologies related to process, recipe and operating procedures. This ontology is where the PSL terms, definitions and their relations are stored.

Concepts relevant to recipes and procedures are defined in process-structural ontology, process-management-and-operation ontology, plant-management-and-operation ontology and in the management and operation generic ontology.

The process-structural ontology can be used to describe the process from the perspective of the sequence of transformations in the processed materials. For example a process described with the Abstract Process Design [7] falls in this category. APD is a technique based on the concept of interconnected process cells. Each cell defines a region where materials are well-mixed and are simulated in a dynamic fashion, including hold-ups and reactions. A particular kind of connection that links cells with reservoirs represents a hypothetical removal of a material component from the cell to the reservoir. The connections between cells assume a flow of material that has the concentration of the source cell.

The process-management-and-operation ontology defines concepts for the construction of equipment-independent activities that describe the chemical process. This is where elements are stored that can be used to represent the procedure section of a general recipe. According to S88, general recipe is defined as "a type of recipe that expresses equipment and site independent process requirements." The procedure information of a general recipe follows a hierarchical structure of three levels, namely process stages, process operations, and process actions. Although the term reaction cell is not defined in the standard, it is specified in the ontology to identify a generic space where the reaction is going to be held, avoiding the use of reactor which adds implicit constraints about the intended equipment.

Operating procedures and the procedural part of batch control recipes are specified with concepts from the plant-management-and-operation ontology. An operating procedure is represented as a multi-level hierarchy of plan steps. On the top of the hierarchy, the procedure is intended to operate the plant or part of it. A procedure is made up of one or more unit procedures that are assigned for a group of equipment or for the whole plant. Examples of unit procedures are 'prepare' and 'react' in 'make PVC'. A unit procedure is made up of operations that are defined to manipulate one or more equipment items such that processing can be safely suspended. In the lower level of the hierarchy, phases act on specific process or control devices. A phase can modify the set point of a controller, send commands to the basic control system to operate a valve, or send a command to other phases. An example of phase is 'activate controller LIC-101,' 'open valve V-101'. Another example is a phase that modifies the state of a multi-stage compressor from off to on. The execution of this phase results in a series of other phases that switch on each stage of the compressor.

The process engineering ontologies are developed using the Stanford Ontology Server [8] that can be utilized remotely from any web browser, allowing extensions carried out by distributed collaborators. The server includes a number of tools for editing, browsing, and consistency checking as well as for report generation. The editing tool (ontology editor) is used to create and update classes, subclasses, attributes, relations and axioms. The editor automatically converts the user's input to a formal representation called Ontolingua. Ontolingua code can then be translated to several different languages, including CLIPS and CORBA IDL. The modeling approach allows the extension and construction of ontologies from already existing ontology libraries, which results in less time spent in modeling other related domains. In addition, the Ontology Server provides mechanisms for validating the syntax and semantics of the definitions. First-order logic sentences that constrain the meaning of the terms defined in the ontology are represented in KIF on which Ontolingua is based.

Operating procedures, recipes and process state transitions are all defined as subclasses of activity. An extension to the original PSL are design rationale and activity agent. Design rationale of an activity describes the arguments that support the plant step. Agent information describes the person, software or device responsible of executing the action or the actual performer that executes the activity-occurrence.

4. OPERATING PROCEDURE EXAMPLE

An operating procedure is represented as a multi-level hierarchy of activities based on a subclass of activity: *operation.* Unlike S88, procedures based on the MDF ontologies have an undetermined number of hierarchical levels. To identify the scope of an operation, associations are used that relate operation to a particular plant system. A plant system which is defined in the plant-structural-ontology represents a single plant item or an assembly of plant items with functional relationships between the parts that performs or can perform a clearly identified processing activity as a whole. A startup procedure for a continuous process can be defined as a series of operations that act upon plant subsystems. An operation that act upon a plant subsystem is then decomposed further into operations that operate paths of equipment. Hierarchical levels exists up to the level where operations manipulate directly specific devices (Figure 2). Compatibility with S88 is possible by extending the plant-system ontology with a relation that links a plant system to the process cell, unit, equipment module or control module instances. Similarly, an extension to the plant-management-and-operation-dimension is possible that specifies a relation of operation to the procedure, unit procedure, S88 operation and phase instances.

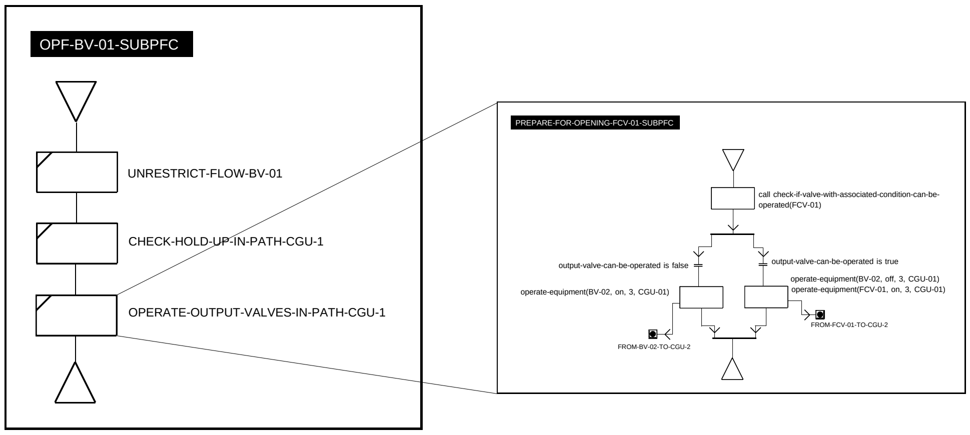
OPF-BV-01-SUBPFC
UNRESTRICT-FLOW-BV-01
CHECK-HOLD-UP-IN-PATH-CGU-1
OPERATE-OUTPUT-VALVES-IN-PATH-CGU-1
PREPARE-FOR-OPENING-FCV-01-SUBPFC
call check-if-valve-with-associated-condition-can-be-operated(FCV-01)
output-valve-can-be-operated is false
output-valve-can-be-operated is true
operate-equipment(BV-02, on, 3, CGU-01)
operate-equipment(BV-02, off, 3, CGU-01)
operate-equipment(FCV-01, on, 3, CGU-01)
FROM-FCV-01-TO-CGU-2
FROM-BV-02-TO-CGU-2

Figure 2. Operation decomposition in the startup operating procedure of the plant system CGU-1

The p-doss prototype that implements a simulation-based approach for operating procedure generation was developed in order to validate and refine the ontologies. Implemented in G2, p-doss has been successfully tested on an HDS industrial process in which initial startup, startup after turnaround, partial, and total shutdown are simulated [9]. The user constructs detailed operating procedure interactively by supplying an incomplete operating procedure that specifies conditions that must be satisfied before actually opening or closing a control valves. For example, the following is necessary to avoid damage to the catalyst: If PIC-06 $\geq$ 75 Kgf/cm^{2}G and TY-28 $\geq$ 175 °C then conclude that the condition-for-open of FCV-01 is true. Such information is represented with the use of the *probably-changes* concept of the PSL ontology.

5. GENERAL RECIPE REPRESENTATION

The processing-step concept defined in the process-management-and-operation ontology is the main building block for general recipe construction. Equipment requirements such as design pressure are represented as preconditions (in PSL a precondition is a fluent that holds before an activity occurrence). General recipe procedures, process stages, process operation and process action can be represented with the processing-step concept (Figure 3). With the ontology extensions, chemists can store general recipes in an electronic format that allows for consistency checking with master and control recipes not mentioning the benefits in the whole recipe management process.

6. CONCLUSIONS

A common representation of plans opens a number of possibilities to integrate design, supply chain management, enterprise-level management and control systems. Hence, specific actions can execute in accordance with their global, longer-term goals for production, maintenance, startup, shutdown and safety. Recently, PSL has been accepted as a *new work item* (ISO 18629) within the ISO TC 184 subcommittee of the International Organization for Standardization that defines industrial data standards for industrial automation systems and integration. Further work is needed, hopefully with the participation of process industry, to define a common ontology for the exchange of process, recipe and procedure information. Collaboration with the PSL developers may ensure that such ontology can be proved to be useful in the process engineering domain.

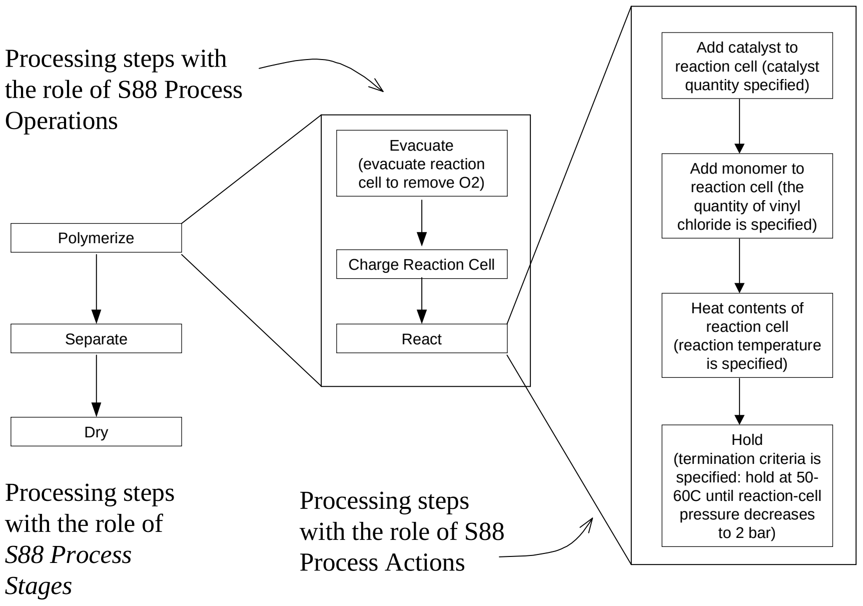
Processing steps with the role of S88 Process Operations
Polymerize
Separate
Dry
Processing steps with the role of *S88 Process Stages*
Evacuate (evacuate reaction cell to remove O2)
Charge Reaction Cell
React
Processing steps with the role of S88 Process Actions
Add catalyst to reaction cell (catalyst quantity specified)
Add monomer to reaction cell (the quantity of vinyl chloride is specified)
Heat contents of reaction cell (reaction temperature is specified)
Hold (termination criteria is specified: hold at 50-60C until reaction-cell pressure decreases to 2 bar)

Figure 3. Processing steps as building blocks for general recipe elements

REFERENCES

1. Parshall, J. and L. Lamb, Applying S88 – Batch Control from a User's Perspective, International Society for Measurement and Control (2000)
2. R. Batres, and Y. Naka, Process Plant Ontologies based on a Multi-dimensional Framework. Proceedings of the Fifth International Conference on Foundations of Computer-Aided Process Design, AIChE Symposium Series, No. 323 (1999)
3. T. Berners-Lee, J. Hendler, and O. Lassila, The Semantic Web. Scientific American, May (2001)
4. C. Schlenoff, M. Gruninger, F. Tissot, J. Valois, J. Lubell, J. Lee, The Process Specification Language (PSL): Overview and Version 1.0 Specification, National Institute of Standards and Technology, Gaithersburg, MD (2000).
5. M. Genesereth, R. Fikes, Knowledge Interchange Format (Version 3.0) - Reference Manual, Computer Science Dept., Stanford University, Stanford, CA. (1992)
6. R. Batres, and Y. Naka, Process Plant Ontologies based on a Multi-dimensional Framework, AIChE Symposium Series, No. 323 (1999)
7. N. Shah, N. J. Samsatli, M. Sharif, J. Borland, and L. Papageorgiou, Modeling and Optimisation for Pharmaceutical and Fine Chemical Process Development, AIChE Symposium Series, No. 323 (1999)
8. A. Farquhar, R. Fikes, J. Rice, Tools for Assembling Modular Ontologies in Ontolingua. Technical Report KSL-97-03, Stanford University, Knowledge Systems Laboratory (1997)
9. R. Batres-Prieto, A multi-dimensional formalism and its implementation in a concurrent engineering environment, PhD Thesis, Tokyo Institute of Technology (1999)