

# A Formalism and Method for the Automated Synthesis of Executable Process Models from SME-authored Process Diagrams

José Manuel Gómez-Pérez  
iSOCO S.A.  
Madrid, Spain  
jmgomez@isoco.com

Oscar Corcho  
Universidad Politécnica de Madrid  
Boadilla del Monte, Madrid, Spain  
ocorcho@fi.upm.es

Michael Erdmann  
Ontoprise GmbH  
Karlsruhe, Germany  
Erdmann@ontoprise.de

## ABSTRACT

Enabling Subject Matter Experts (SMEs) to formulate knowledge without the intervention of Knowledge Engineers (KEs) requires providing SMEs with methods and tools that abstract the underlying knowledge representation, allowing SMEs to focus on the modeling activities. However, automatically bridging the gap between SME-authored models and their internal representation is not an easy task, especially in the case of complex knowledge types like processes, where aspects like frame management, data, and control flow need to be addressed. In this paper, we present a process representation formalism and method for automatically grounding SME-authored process models in the form of process diagrams into a particular representation language, supporting process representation and reasoning.

## Categories and Subject Descriptors

I.2.4 Knowledge Representation Formalisms and Methods – representations (procedural and rule-based), I.2.8 Problem Solving, Control Methods, and Search – plan execution, formation, and generation.

## General Terms

Algorithms, Measurement, Performance, Design, Human Factors, Languages, Theory, Verification.

## Keywords

Process knowledge representation, SMEs, PSMs, F-logic.

## 1. INTRODUCTION

Enabling Subject Matter Experts (SMEs) to model processes by themselves without the intervention of Knowledge Engineers (KEs) is a complex problem that needs to be addressed from a multidimensional perspective in order to: i) provide the required knowledge artifacts to acquire process knowledge and ii) develop usable tools enabling SMEs to exploit such artifacts. To this purpose, our work has focused on producing the following models, methods and tools:

1. A process metamodel, which provides the terminology necessary to express process entities in scientific domains and the relations between them.
2. A library of Problem Solving Methods [4], which provides high-level, reusable abstractions for process representation and reasoning strategies.

3. A graphical modeling and reasoning environment, which leverages the process metamodel and the PSM library to enable SMEs to model processes.
4. A formalism and method for the automatic synthesis of executable process models from SME-authored process diagrams.

While we presented the first three outcomes in [3], herein we focus on the fourth. More details about the overall approach can be found in [2].

## 2. REPRESENTING AND REASONING WITH PROCESS KNOWLEDGE

We consider four main types of process reasoning to be supported by the formalism: i) reasoning about process entities, ii) intermediate results, iii) process stages, and iv) process preconditions. For example, the multiple-choice question below, selected from Advanced Placement (<http://apcentral.collegeboard.com>) exams in Biology, illustrates the third type of process reasoning.

Which part of the animal cell is required only in the first stage of mitosis and what is the name of such stage?

- a. chromatin and prophase
- b. chromatid and prometaphase
- c. centromere and anaphase
- d. plasma membrane and telophase

In our formalism, a process consists of a set of process actions, connected in the form of a directed graph, with pre and post conditions whose evaluation both determines the flow of data between process steps and controls the order in which such actions are executed. We define the *pre* and *post states* of an action respectively as the content of the process frame immediately before and after its execution. The pre state contains all the process resources in the knowledge base that serve as inputs to the action, while the post state contains the outcomes of its execution, obtained by operating on the contents of the pre state.

At modeling time, our code generation method automatically synthesizes sound and complete executable code in the form of F-logic rules associated to each action in a process model. F-logic i) provides a single entry point for reasoning, supporting the different knowledge types involved in particular questions, ii) enables the use of rule knowledge for reasoning within processes, and iii) keeps introspective properties for retrieval of meta-information about processes, like subprocesses and intermediate

process results. Process rules manage the process frame [5] in order to support data and control flow and can be classified as follows: i) **setup rules**, which take the relevant portion of knowledge from the overall knowledge base, ii) **transition rules**, which describe the transformation of inputs into outputs, and iii) **precedence rules**, which transfer the output of actions to their successors. We optimize performance by avoiding second order reasoning and well-founded semantics evaluation mode. Next, we show the sample F-logic code of a transition rule corresponding to a muscle contraction process:

```
FORALL m, e, j
j: jump@postState(muscleContraction) AND
j: OUTPUT@postState(muscleContraction) AND
muscleContraction[PROVIDES -> j] @postState(muscleContraction)
<-
m:muscle @preState(muscleContraction) AND
m:TOOL@preState(muscleContraction) AND
m[IS_USED_BY -> muscleContraction]@preState(muscleContraction) AND
e:energy@preState(muscleContraction) AND
e:RESOURCE@preState(muscleContraction) AND
e[IS_CONSUMED_BY -> muscleContraction]@preState(muscleContraction).
```

### 3. EVALUATION

This work was evaluated by an independent team in the context of project Halo. Six SMEs formulated knowledge on the selected syllabi for the domains of Chemistry, Biology, and Physics, and tested reasoning with it. The quality of the resulting knowledge bases was determined by test sets created by the SMEs themselves through the testing & debugging tool in the system in order to check that their process models actually behaved as expected. 82% of the process models were correct. In all cases, process models were formulated by SMEs without intervention of KEs and only required initial training and sporadic support in the utilization of the tools.

The process modeling environment was rated by SMEs with an average of 64.5 out of 100 in the System Usability Scale [1]. As to utility, SMEs modeling process knowledge rated the approach with an average of 3 points out of 4, especially in the domain of Biology. Additionally, personal interviews with SMEs showed a high degree of satisfaction, with comments like “*It makes the representation of biological models easier*” (SME2) and “*The modeling of processes is very useful. It must be possible to ask questions about the various states of a process. And asking questions with T&D worked okay*” (SME3).

We studied the effects of the application of the optimizations described in section 6 to the F-logic code resulting from the process models formulated by the SMEs. We measured response times (Table 1) of a sample of ten queries uniformly distributed across the four reasoning types described in section 2. These queries were executed against the Biology knowledge base produced by SME3, which contained the largest sample of process knowledge produced by the SMEs in the evaluation, with three different configurations of the F-logic reasoner OntoBroker, combining different uses of well-founded evaluation and second-order reasoning. C0 is the most generic configuration, with the well-founded evaluation mode enabled and concept and attribute names ground disabled. C1 and C2 correspond to the optimization methods described in the previous section. C1 aims at increasing performance with respect to C0 by enabling concept and attribute

names ground while C2 extends C1 by additionally disabling well-founded evaluation.

The results of executing this query set with the three different configurations are shown in Table 1 (values equal to 0 stand for queries with response times lower than 1ms) shows an average performance improvement of 25% for C1 and almost 30% for C2. The main reason is that concept and attribute names are ground as in C1. C2, an extension of C1 that also disables well-founded evaluation mode, adds in this case little performance gain since the code generation mechanism already produced most of the code in well-stratified form, hence reducing the need of well-founded semantics.

Query	with respect to configuration C0					
	C0	C1		C2		
SME3-q0	31	1,00	0	0,00	16	0,52
SME3-q1	63	1,00	16	0,25	16	0,25
SME3-q2	31	1,00	16	0,52	16	0,52
SME3-q3	47	1,00	16	0,34	16	0,34
SME3-q4	15	1,00	0	0,00	0	0,00
SME3-q5	32	1,00	16	0,50	0	0,00
SME3-q6	203	1,00	219	1,08	234	1,15
SME3-q7	63	1,00	31	0,49	31	0,49
SME3-q8	47	1,00	31	0,66	16	0,34
SME3-q9	62	1,00	32	0,52	16	0,26
SME3-q10	203	1,00	218	1,07	203	1,00
Average	79,7	1,00	59,5	0,75	56,4	0,71
Median	47	1,00	16	0,34	16	0,34
Min	15	1,00	0	0,00	0	0,00
Max	203	1,00	219	1,08	234	1,15

< 1 - faster  
= 1 - same time as config C0  
> 1 - slower

Table 1: C1 and C2 compared with reference C0

### 4. ACKNOWLEDGMENTS

This work has been funded by Vulcan Inc. as part of Ontoprise’s DarkMatter project within the Halo project (<http://www.projecthalo.com>).

### 5. REFERENCES

- [1] Brooke, J. (1996) *SUS: a "quick and dirty" usability scale*. In P. W. Jordan, B. Thomas, B. A. Weerdmeester & A. L. McClelland (eds.) *Usability Evaluation in Industry*. London.
- [2] Gómez-Pérez, J.M. *Acquisition and Understanding of Process Knowledge using Problem Solving Methods*. Studies on the Semantic Web vol. 7. AKA Verlag – IOS Press. ISBN: 978-3-89838-639-5. June 2010.
- [3] Gómez-Pérez, J.M., Erdmann, M., Greaves, M. *Applying Problem Solving Methods for Process Knowledge Acquisition, Representation, and Reasoning*. KCAP 2007.
- [4] McDermott, J. *Preliminary steps towards a taxonomy of problem-solving methods*. In Marcus, S., editor, *Automating Knowledge Acquisition for Expert Systems*, pages 225-255. Boston, Kluwer.
- [5] Pylyshyn, Z.W. *The Robot’s Dilemma: The Frame Problem in Artificial Intelligence*. Norwood, 1987.