A Framework and Computer System for Knowledge-level Acquisition, Representation, and Reasoning with Process Knowledge

José Manuel Gómez-Perez[1], Michael Erdmann[2], Mark Greaves[3], Oscar Corcho[4], and Richard Benjamins[5]

[1]Intelligent Software Components (iSOCO) S.A.
[2]Ontoprise GmbH
[3]Vulcan Inc.
[4]Universidad Politécnica de Madrid
[5]Telefónica I+D

ABSTRACT

The development of knowledge-based systems is usually approached through the combined skills of software and knowledge engineers (SEs and KEs, respectively) and of subject matter experts (SMEs). One of the most critical steps in this task aims at transferring knowledge from SMEs' expertise to formal, machine-readable representations, which allow systems to reason with such knowledge. However, this process is costly and error prone. Alleviating such *knowledge acquisition bottleneck* requires enabling SMEs with the means to produce the target knowledge representations, minimizing the intervention of KEs. This is especially difficult in the case of complex knowledge types like processes. The analysis of scientific domains like Biology, Chemistry, and Physics uncovers: i) that process knowledge is the single most frequent type of knowledge occurring in such domains and ii) specific solutions need to be devised in order to allow SMEs to represent it in a computational form. We present a framework and computer system for the acquisition and representation of process knowledge in scientific domains by SMEs. We propose methods and techniques to enable SMEs to acquire process knowledge from the domains, to formally represent it, and to reason about it. We have developed an abstract process metamodel and a library of Problem Solving Methods (PSMs), which support these tasks, respectively providing the terminology for SME-tailored process diagrams and an abstract formalization of the strategies needed for reasoning about processes. We have implemented this approach as part of the DarkMatter system and formally evaluated it in the context of the intermediate evaluation of Project Halo, an initiative aiming at the creation of question answering systems by SMEs.

1. INTRODUCTION

Building knowledge-based systems is an activity that has been traditionally carried out by a combination of software and knowledge engineers and of subject matter experts (SMEs), also known as domain experts. Software engineers (SEs) are focused on architectural and user interface issues related to the development of software. Knowledge engineers (KEs) are focused on knowledge acquisition and representation tasks, with the aim of building the required knowledge bases. For these tasks, they normally work in collaboration with SMEs, who act as repositories of domain knowledge to a large extent. The combination of KEs and SMEs is feasible for a number of domains. However, it has two main drawbacks, first characterized as the *knowledge acquisition bottleneck* by Feigenbaum in 1977: i) it is costly and ii) it can be error prone, especially in complex domains.

A large amount of work in knowledge-based systems in the past three decades has concentrated on providing frameworks and tools that support the collaboration of KEs and SMEs with the goal of alleviating the knowledge acquisition bottleneck. Despite such work, existing knowledge acquisition tools are still not effective and intuitive enough to allow SMEs to capture the knowledge from a domain by themselves.

Among the different types of knowledge that can be used in knowledge-based systems, in our work we focus on the particular case of *process knowledge*. Process knowledge is one of the most widely used but also complex types of knowledge across domains, posing important challenges for knowledge acquisition. A process can be considered as a special concept which encapsulates such things as preconditions, results, contents, actors, or causes and can be defined as "a naturally occurring or designed sequence of changes of properties of a system or object".[1] For example, consider a complex chemical reaction comprising several steps, with different inputs and outputs, where reasoning about what would happen at a certain stage if a previous one was suppressed is required. Processes also relate to the sequence of operations and involved events, taking up time, space, expertise or other resources, which lead to the production of some outcome.

Our motivation to focus on process knowledge stems from the fact that current approaches to knowledge representation do not suffice at representing this kind of information. Several approaches have been proposed from different areas and perspectives, including the following: i) knowledge acquisition and representation languages, e.g., OWL[2], OCML (Motta, 1998), F-Logic (Kifer et al., 1995), and KARL (Fensel et al., 1998); ii) process-specific representation and reasoning languages, e.g., PSL (Bock and Grüninger, 2005) and SPARK-L (Morley and Myers, 2004); iii) semantic web service ontologies, e.g., WSMO[3] and OWL-S[4]; and iv) process

[1] http://en.wikipedia.org/wiki/Glossary_of_systems_theory

[2] http://www.w3.org/TR/owl-features

[3] http://www.wsmo.org

[4] http://www.w3.org/Submission/OWL-S

specification and execution languages, e.g., BPEL[5]. However, while expressive in terms of workflow constructions and reasoning capabilities, such approaches either suffer from high complexity or low abstraction capabilities that hinder their use, especially by SMEs. Thus, while fundamental for the construction of knowledge-based and workflow systems, further solutions are required that can be used to address the problem for the process knowledge case.

Furthermore, we aim at enabling SMEs themselves to model and reason with process knowledge without intervention of KEs. In this context, not only is it necessary to deal with the intricacy of process knowledge but also with the lack of knowledge engineering skills by SMEs. Given all this complexity, the mechanisms required for acquiring and reasoning with process knowledge must be flexible and reusable, enabling their exploitation across several scientific and non-scientific domains (ecology, engineering, business, etc.) with as little effort as possible.

Thus, our main objective is to produce the means required to enable SMEs to acquire, formally represent, and reason about processes without the intervention of KEs. In order to describe our work towards achieving such objective, this article is structured around three main conceptual blocks: i) the creation of knowledge artifacts that support the acquisition of process knowledge, ii) the development of usable tools allowing SMEs to exploit such artifacts, and iii) the formal evaluation of the whole approach with real SMEs from scientific domains. The accomplishment of the tasks comprised by these blocks has resulted in the following models, methods and tools, which will be presented in the article:

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* (McDermott, 1988), which provides high-level, reusable abstractions for process representation, *and the method used for its development*, which facilitates filtering and producing such abstractions from amongst all the other knowledge types in the available documental sources.
3. A *graphical process modeling and reasoning environment*, which applies the process metamodel and the PSM library in order to enable the creation and editing of user-tailored process diagrams, without intervention of KEs.
4. A method for the *automatic synthesis of executable process models from SME-authored process diagrams*, supported by an underlying representation and reasoning formalism.

Problem Solving Methods (PSMs) are central to this work. They were conceived as domain-independent, reusable knowledge strategies that can be applied in different application domains to solve conceptually similar problems in terms of the goals to be achieved and the type of knowledge required (Fensel and Benjamins, 1998). PSMs have been traditionally used in knowledge engineering in three main ways: i) for knowledge acquisition, as guidelines to acquire knowledge that allows solving problems, ii) for reasoning, and iii) analytically, for describing the main rationale behind a reasoning process. In this work, we report on the first use of PSMs for the specific case of acquiring process knowledge.

[5] http://www.oasis-open.org/committees/wsbpel

Through the application of these models, methods, and tools to the problem of acquiring process knowledge by SMEs, we pursue four main outcomes, which can be preliminarily introduced here as follows:

1. Higher quality and less costly process knowledge bases, through empowering SMEs and taking KEs out of the process knowledge acquisition loop.
2. Reduced complexity of acquiring process knowledge by SMEs through the use of PSMs as domain-independent, reusable abstractions of domain-specific processes.
3. Keeping acquisition of process knowledge at the knowledge level (Newell, 1982), through an underlying process knowledge representation formalism transparent to SMEs.
4. Flexible and reusable mechanisms for acquisition and reasoning with process knowledge by SMEs, maximizing the application of the approach across several domains with little effort.

The remainder of the article is structured as follows. Sections 2 to 4 focus on the first of the abovementioned conceptual blocks (knowledge artifacts), presenting the knowledge structures proposed in order to support SMEs in the acquisition of process knowledge. In particular, section 2 describes the analysis of the different knowledge types appearing in the target domains and explains why process knowledge is one of the most relevant types. Section 3 focuses on the process metamodel. To finalize this block, section 4 focuses on the PSM library for the acquisition of process knowledge, including the method used for its development.

The second block (usable tools) is treated in section 5, where we describe how the previous knowledge structures can be articulated in a real system, a process editor integrated in the DarkMatter (Deep Authoring, Answering and Representation of Knowledge by Subject Matter Experts) system of Project Halo, [6] which allows SMEs to model processes at the knowledge level. In this section, we also introduce briefly the underlying formalism for process knowledge representation, which is described in more detail in (Gómez-Pérez, 2009). Section 6 deals with the third block (formal evaluation), and presents the evaluation of the approach in the context of Project Halo and its value for actual SMEs. Finally, section 7 provides conclusions based on the analysis of the evaluation results and section 8 discusses the contributions of this work and proposes directions for future work.

2. MOTIVATION: THE PROCESS KNOWLEDGE TYPE

In (Friedland et al., 2004), KEs directly encoded parts of a Chemistry textbook into formal knowledge representation languages, proving the practicality of representing sufficient knowledge for a computer to solve scientific problems at a level comparable to AP[7] students. However, this effort confirmed the problems derived from the knowledge acquisition bottleneck. The cost of encoding one textbook page proved impractically high (approximately $10,000 per textbook page) and the evaluation of the system showed evidence that an important part of the system failures reflected insufficient expertise of domain knowledge by KEs.

[6] www.projecthalo.com

[7] Advanced Placement (apcentral.collegeboard.com)

The outcome of such initiative concluded that addressing the knowledge acquisition bottleneck requires enabling SMEs themselves to represent knowledge from the target domains. This objective implies three major challenges. First, the system needs to possess sufficient problem solving power to solve scientific problems. Second, it must be able to explain its answers in a human-understandable and domain-grounded way. Third, the interface must be user-friendly, yet allow SMEs to formulate and exploit the large, complex body of scientific knowledge, after adequate training.

In order to estimate the effectiveness of existing technologies in addressing these challenges in terms of knowledge representation, knowledge formulation and question formulation, joint teams of SMEs and KEs worked with a representative corpus of 755 AP questions from the target domains of Chemistry, Biology, and Physics (Valente et al., 2004). As a result, and after two main phases comprising a platform independent domain analysis and a platform specific knowledge engineering analysis, the following knowledge types were identified: classification (CLS), comparison (CMP), factual knowledge (FACT), inference rules (RULE), mathematics (MAT), process knowledge (PCS), causality (CAUS), procedural (PROC), basic data structures (DAT), tables (TAB), part-whole (PWR), spatial (SPACE), temporal (TIME), representational (TRANS), experimental (EXP), non functional (NF), graphic (GRA), and under-specified knowledge (US).

Most of these knowledge types occur across the three domains with varying fractions. Figure 1 shows the number and percentage of questions from the three domains dealing with each knowledge type, as well as the average across all domains. Additionally, Figure 2 shows the overall average ranking of the different knowledge types. It can be appreciated in the figures that a same question may deal with different knowledge types simultaneously. Consequently, the summation of the different knowledge types across the corpus of AP questions exceeds 100%.

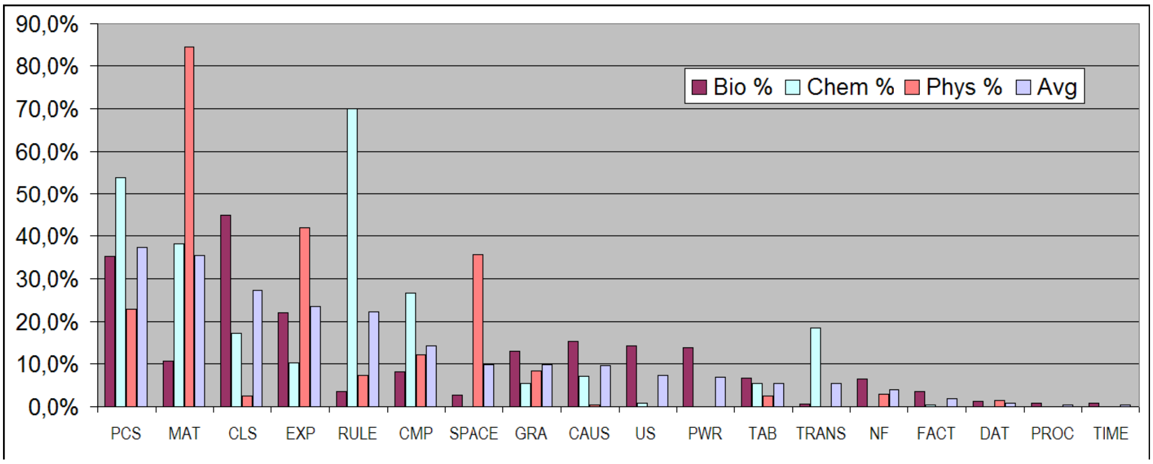


Figure 1: Per-domain distribution of knowledge types

Overall, process knowledge is the most frequently used type of knowledge for answering AP questions, with 37% average across the three domains. This is even clearer if we go through the individual domains. In Chemistry, process knowledge is the most important knowledge type, occurring in 53% of all the Chemistry questions. It also scores second in Biology, with 35%, only after classification knowledge. Finally, process knowledge is the fourth knowledge type in Physics, with 22%, after mathematical, experimental, and spatial knowledge.

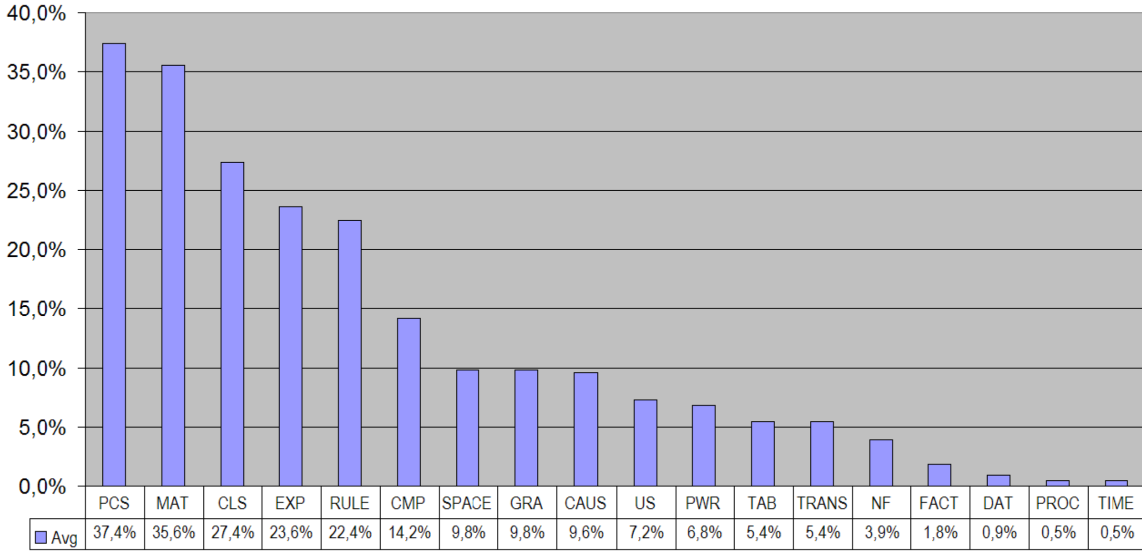


Figure 2: Average distribution of knowledge types

As shown above, one of the outcomes of this analysis is the preeminence of process knowledge against the remaining knowledge types identified. The second outcome of the analysis in with respect to process knowledge was platform-specific. The platform-specific analysis resulted into a characterization of how the available technologies would address the challenges posed by such domains in terms of knowledge engineering tasks. In the case of process knowledge, the analysis concluded that such technologies would not support the acquisition of process knowledge from the domains by SMEs, hence requiring new solutions for the process knowledge case. The analysis especially advised i) the development of a high-level process ontology or core theory for representing reoccurring models in the domains and ii) to investigate new methods and tools to simplify formulation and access to process knowledge by SMEs.

In the following sections, we detail our approach towards designing and implementing specific solutions for acquisition and reasoning with process knowledge by SMEs. The remaining types of knowledge fall out of the scope of this article.

3. THE PROCESS METAMODEL

The objective of the process metamodel is to provide SMEs with the minimal building blocks, in the form of a process-specific vocabulary, to formulate processes. The process metamodel reuses parts of pre-existing process ontologies and builds on them in order to provide the terminology necessary to express process entities like agents, actions, resources, etc. and the relations between them.

3.1 Reused Process Ontologies

Following ontology engineering methodological guidelines (Fernández-López et al., 1997), we reused several process ontologies, among which some of the most relevant ones are the Enterprise Ontology (Uschold et al., 1998), the Toronto Virtual Enterprise project (TOVE)[8] ontologies, and the GuideLine Interchange Format Ontology (Ohno-Machado et al., 1998) (GLIF).

The Enterprise Ontology and its extension TOVE define collections of terms and definitions relevant to business processes. These ontologies deal with the following main areas:

- **Activity** captures the notion of anything that involves some action. The concept of activity is closely connected to the notion of doer, i.e., the agent that performs or participates in the action. Another entity closely related to activity is resource, which can be consumed by the action or just required, e.g., as a lookup resource. Activities can have certain duration and scheduled time of application, and also show effects on other entities. Additionally, activities can be aggregated to form more complex activities composed of a series of subactivities. If activities have an intended purpose its specification is called a plan.
- **Organization** contains candidate doers, i.e., potential actors or agents. They can be classified as either legal entities or organizational units.
- **Strategy** is defined as a plan to achieve a high-level purpose.
- **Marketing** includes concepts like *sale*. A sale is an agreement between two legal entities for the exchange of a product for a sale-price.

Among these, both the *activity* and the *organization* parts have provided valuable contributions to the process metamodel, specifically to concepts representing agents and resources in the process resource section, to the process action section, and to action, agent, and resource relations in the process relation section (Figure 3). The rest of these ontologies are too specific to the business domain, and therefore inconsistent with our aim to maximize domain-independence and reusability of the process metamodel.

GLIF is a framework for modeling biological processes partially based on the workflow model of the Workflow Management Coalition (WfMC[9]). In this case, we extracted a number of simple workflow primitives and incorporated them in the process metamodel, namely different types of iterative actions and forks.

3.2 Conceptual Model

Figure 3 shows the taxonomy of the process entities contained in the process metamodel. The main entities contained in this model and their connections are described next.

[8] http://www.eil.utoronto.ca/enterprise-modelling/tove

[9] http://www.wfmc.org

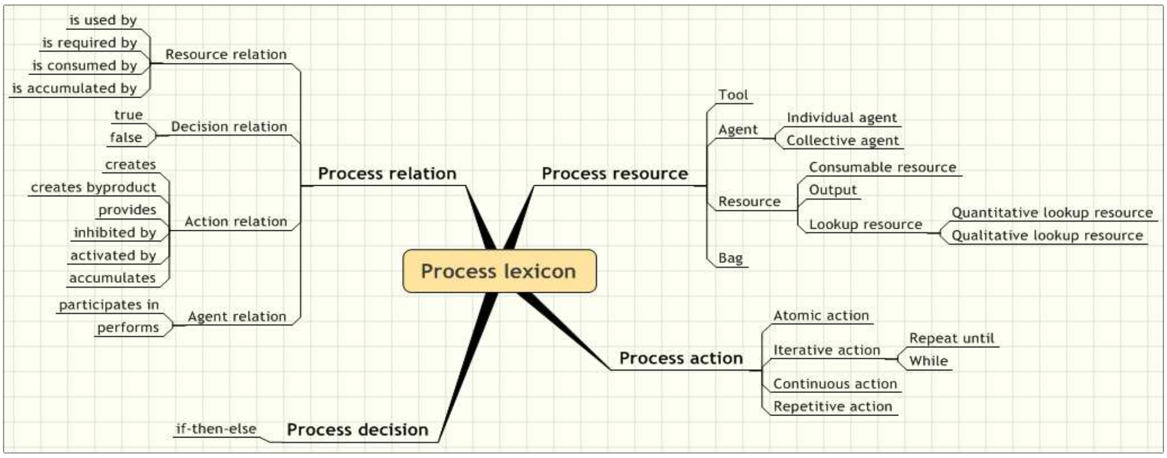


Figure 3: Conceptual diagram of the process metamodel entities

Process resources

This category contains all the entities that can be used as resources within a process: *tools*, used by a given agent to perform an *action*, *bags* used to group other entities, implicitly representing partonomic hierarchies, and, in general, *resources*, used by agents participating in activities, as well as *agents* themselves. An agent can be classified as an *individual agent*, e.g., bee, or a *collective agent*, e.g., swarm. A resource can be classified as a *consumable resource*, if it can be spent during a process or as a *lookup resource* if it is used as a recipient of information. In this case, a resource can be *quantitative*, whenever it is numerable, or *qualitative* in other case. Resources can be termed as *output* when resulting from the execution of a process.

Process resources are also called *roles* (Wielinga et al., 1992) according to the PSM nomenclature. Roles serve two purposes, first they act as a container for domain concepts and, second, as a pointer to the types of domain concepts that can play this role. Domain concepts may play different roles during reasoning either in the same or across different processes. For example, *water* can take the role of a consumable resource during the process of a precipitation reaction, but on the other hand another SME might consider it as the agent performing the reaction. Figure 4 shows the graphic representation of the main process resources in the process metamodel.

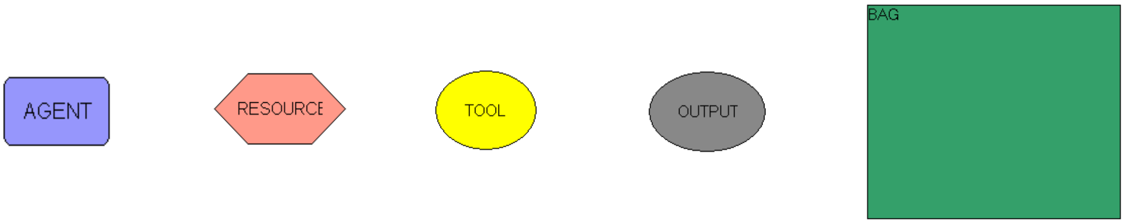


Figure 4: Main types of process resources

Roles can be divided into *static* and *dynamic* roles (Wielinga et al., 1992). Static roles contain concepts that are persistent across the reasoning process. Dynamic roles contain concepts that change during the reasoning process. Dynamic roles characterize the process because they are constantly manipulated by the process in which they are used. In DarkMatter both static and dynamic roles are supported. Finally, in actual, working systems, users need to contextualize the different roles in terms of the domain. In DarkMatter, this is done by mapping from domain concepts into the roles of an instantiated process.

Process relations

This category contains process-level relations which can take place in a given process between the different actors participating in it. These relations can happen both between resources and actions, actions and resources, and between actions and other actions. A resource can be *used, required, consumed,* and *accumulated* by an action. Actions can succeed a previous action if a given condition is *true* or *false*. Actions can *create, provide* or *accumulate* a resource as a consequence of its execution. Additionally, actions can be *inhibited* or *activated* by an agent, and agents can *participate in* or *perform* actions.

Process decisions

A *decision* (Figure 5) is a workflow construct resembling forks depending on a conditional expression. These forks are used to explicitly create conditional precedence relations between pairs of actions. Depending on the results of evaluating such condition, the precedence relation will be enabled *(true)* or not *(false)*.

Figure 5: Conditional fork

Process actions

Process actions are classified as *atomic, iterative, continuous,* or *periodic.* Atomic actions consist of the transactional execution of activities, e.g., binding two amino acids in Biology, while iterative actions are executed repeatedly while or until a certain condition holds, e.g., the iterative process of RNA synthesis from DNA templates where an enzyme follows a DNA template until a termination sequence is detected. On the other hand, continuous actions are simple actions that have a prolonged duration in time, e.g., "A piece of solid calcium is heated in oxygen gas…". Finally, periodic actions refer to actions which happen repeatedly, provided a given amount of time or events, e.g., the oscillation of a pendulum. The definition of actions has been inspired by that of *activities* as defined in the Enterprise Ontology. On the other hand, workflow constructs used to represent iterative, continuous, and periodic actions have been adapted from GLIF. Figure 6 shows the graphic representation of the main types of actions.

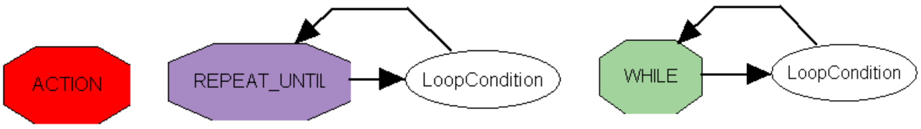


Figure 6: Main types of process actions4.

4. PROBLEM SOLVING METHODS FOR THE ACQUISITION OF PROCESS KNOWLEDGE

We approach processes as special types of problems and PSMs as the way to represent and reason with them. To this purpose, we have produced a PSM library which i) provides SMEs with modeling guidelines that simplify the representation of domain-specific processes and ii) provides the means to reason about and solve process-related problems. Next, we provide an extension of existing modeling frameworks based on PSMs in order to support the process knowledge case and describe the method followed to build our PSM library in the resulting framework. Finally, we extensively describe the domain-independent library of PSMs produced as a result of applying such method.

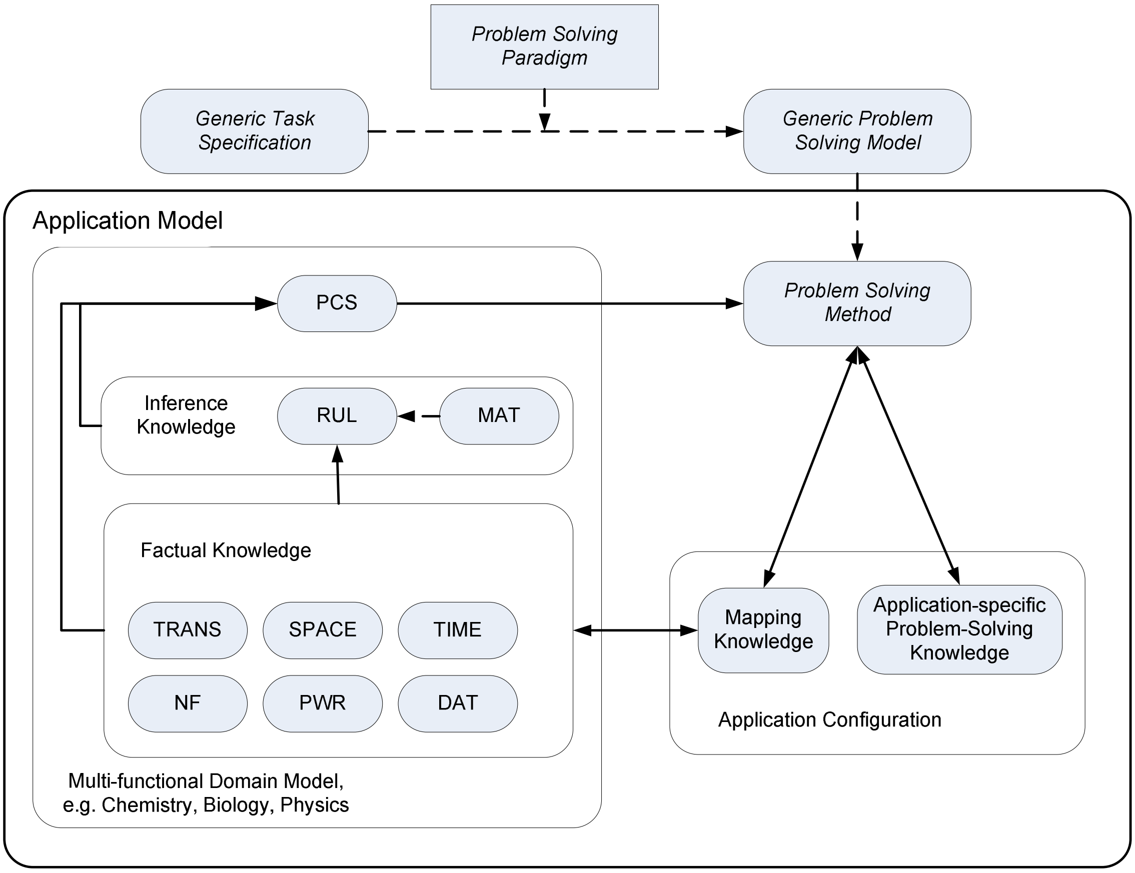
4.1. A PSM Modeling Framework for Processes

There is a considerable number of existing PSM libraries that have been previously used in knowledge-based systems (Chandrasekaran, 1986; Eriksson et al., 1995; McDermott, 1988; Breuker et al., 1987; Benjamins, 1995). For example, the work described by the latter provides a model of the diagnostic problem solving process, using approaches for modeling problem-solving behavior such as those described in (Breuker et al., 1987; Chandrasekaran, 1986; Steels, 1990). As a result, KEs receive support and guidance in constructing models of diagnostic reasoning and it is possible for KEs to combine different approaches in a single diagnostic system, with alternatives for realizing the same tasks.

Modeling frameworks like TMDA (Motta, 1999) and CommonKADS (Schreiber et al., 2000) support the development of such PSM libraries by separating problem-solving behavior from domain-specific knowledge. According to such frameworks, application models need to identify different but interfaced layers of knowledge. CommonKADS proposes three generic types of components: *domain*, *task*, and *inference* while TMDA distinguishes between four, with an additional layer on *application-specific* knowledge. As a consequence, the resulting knowledge bases only contain factual, domain-specific knowledge, much easier to evolve, while problem-solving knowledge can be kept independently from the domain in order to abstract and favor reusability of the procedural and inference knowledge across different domains and tasks.

Using PSMs for acquiring process knowledge also benefits from this kind of approaches. PSMs provide a thorough analysis of domain-specific tasks and problems as well as well-formed strategies on how to solve these problems by means of abstracting the reasoning processes involved. These modeling frameworks assume that the tasks to be accomplished reside at the meta-level as strategies to solve domain-related problems, like, e.g., in (Marcus et al, 1998). However, they do not contemplate the representation of actual processes. In order to overcome this, we propose a specialization of the TMDA component-based modeling framework (Figure 7) that utilizes PSMs as domain-independent components for the specific case of process knowledge.

Such specialization explicitly represents the different knowledge types detected in the analysis of the domains described in section 2 and, in particular, processes. This allows treating process knowledge as higher-level abstractions, decoupled from all other knowledge types in the domain but interfacing with them in order to consume domain-specific rule and factual knowledge. We exploit the relation between ontologies and PSMs as in (Crubézy and Musen, 2003) to describe domain knowledge bases and PSMs as independent components that can be reused and to mediate knowledge between these two components. Our PSM library can thus be kept reusable for acquiring process knowledge across different purposes and domains.



Problem Solving Paradigm
Generic Task Specification
Generic Problem Solving Model
Application Model
PCS
Problem Solving Method
Inference Knowledge
RUL
MAT
Factual Knowledge
TRANS
SPACE
TIME
NF
PWR
DAT
Mapping Knowledge
Application-specific Problem-Solving Knowledge
Application Configuration
Multi-functional Domain Model, e.g. Chemistry, Biology, Physics

Figure 7: Adapted TMDA modeling framework for the process knowledge type (PCS)

Supporting such approach requires four different types of ontologies: i) a PSM-description ontology, which models the competence of a PSM library, ii) a method ontology specifying input and output roles of the PSMs, iii) a mapping ontology relating domain and PSM ontologies, and iv) a domain ontology containing factual knowledge. In our case, (i) and (ii) reflect into the PSM library for the process knowledge type and the process metamodel and iv) are SME-authored ontologies in the domains of Chemistry, Biology, and Physics. Mappings relating domain and PSM roles are 1:1 relations graphically described by SMEs (see Figure 17 in section 5). The semantics of these mappings is a subsumption of the PSM role by the domain concept or instances against which it is mapped.

4.2. A Method to Build a PSM Library of Process Knowledge

Existing PSM ontologies like the above mentioned focus on problem-solving reasoning process in tasks like, e.g., *diagnosis* and *classification*, but do not properly address domain-level processes. Thus, it is necessary to build a PSM library that informs the extended process modeling framework depicted in Figure 7. The construction of such PSM library for representing and reasoning with processes comprises two major steps:

1. **Identification of domain-specific processes** in the syllabi of Biology, Chemistry, and Physics
2. **Decomposition and abstraction** of the previously identified domain-specific processes into primitive, domain-independent, reusable PSMs.

This *top-down* and *bottom-up* process is grounded in the domains but also maximizes domain-independence, reusability, and composition properties.

Identification of domain-specific processes

The first step consists of detecting domain-specific processes and their definitions from amidst the domain analysis documents. From the 755 AP questions for the three domains studied in the analysis phase, we retrieved, for each process occurring in the portion of the syllabi associated to each question, its actual definition according to the textbooks[10]. This process resulted into approximately 100 different domain-specific processes. The analysis of the characteristics and the affinities between these processes showed that they could be clustered in 4 main categories: *Join, Split, Modify, and Locate*. Figure 8 shows how the processes detected in the syllabus are distributed across these categories.

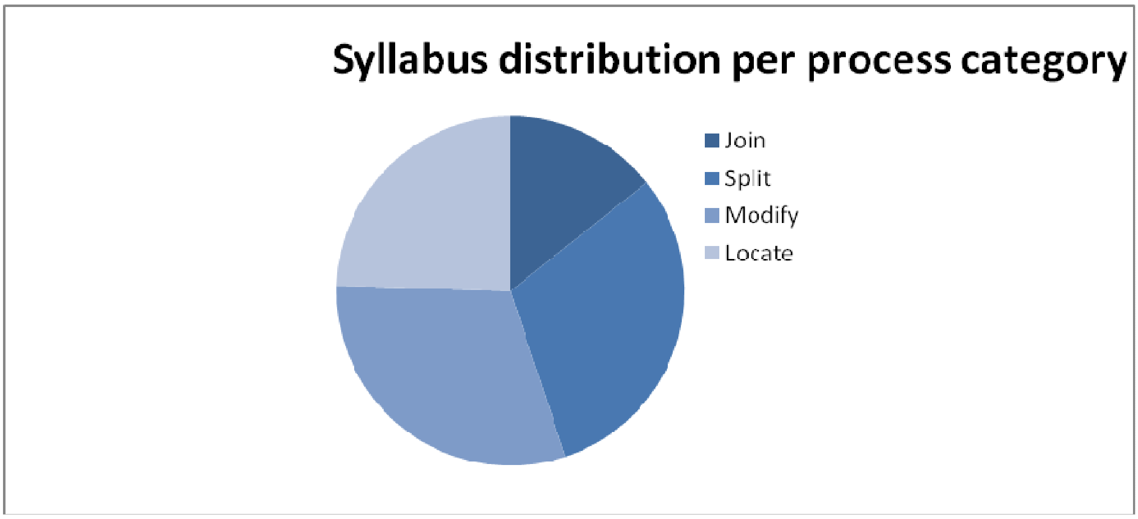
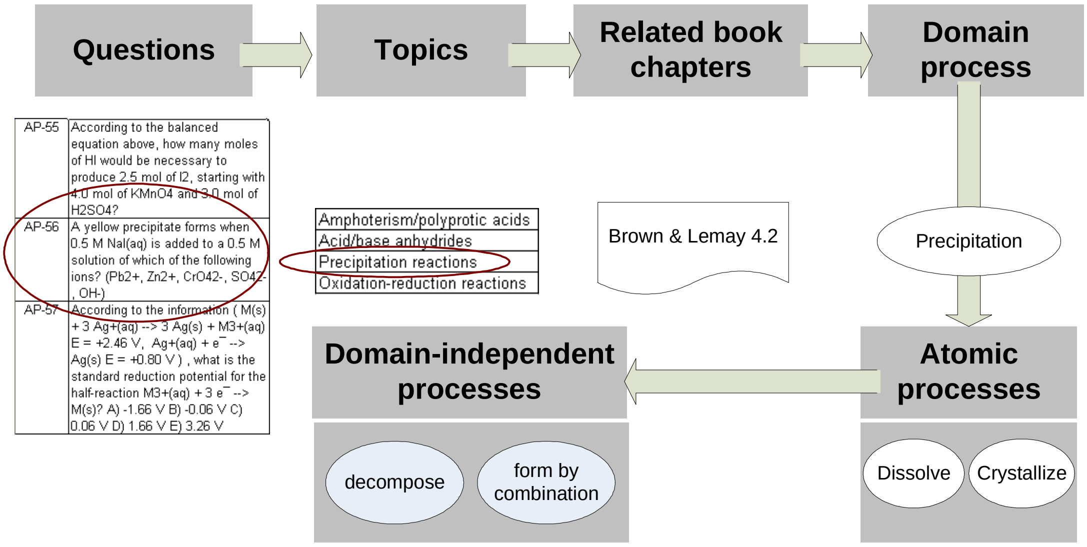


Figure 8: Distribution of process occurrences

Since the goal of this process is to detect all the occurrences of process knowledge in the selected AP questions, we took an incremental approach starting with Chemistry, a priori, according to the analysis (Figure 1), the domain with a larger amount of process knowledge, and then continued with Biology, and finally Physics. The method applied (first part of Figure 9) consisted of the following steps: i) identify the topics related to each question, ii) for each topic, find the specific chapter of the textbook on the target domain dealing with it, and iii) browse the chapter for occurrences of processes, attending to several indicators about their structure, namely preconditions, postconditions, states, actors, inputs, and outputs.

An additional result of this process is the identification of the most frequent verbs used to specify processes and their synonyms. In the next step of this method, *decomposition and abstraction*, this knowledge was used to group the detected domain-specific processes into categories defined by the semantics of these verbs, across the three domains, facilitating the task of creating process abstractions for each of these categories. Such process abstractions eventually led to our PSM library for the process knowledge type (Figure 10).

[10] (Brown et al., 2002; Campbell and Reece, 2001; Serway and Faughn, 2003) were selected as reference textbooks for the Chemistry, Biology, and Physics domain, respectively.



AP-55	According to the balanced equation above, how many moles of HI would be necessary to produce 2.5 mol of I2, starting with 4.0 mol of KMnO4 and 3.0 mol of H2SO4?
AP-56	A yellow precipitate forms when 0.5 M NaI(aq) is added to a 0.5 M solution of which of the following ions? (Pb2+, Zn2+, CrO42-, SO42-, OH-)
AP-57	According to the information (M(s) + 3 Ag+(aq) --> 3 Ag(s) + M3+(aq) E = +2.46 V, Ag+(aq) + e¯ --> Ag(s) E = +0.80 V) , what is the standard reduction potential for the half-reaction M3+(aq) + 3 e¯ --> M(s)? A) -1.66 V B) -0.06 V C) 0.06 V D) 1.66 V E) 3.26 V

Amphoterism/polyprotic acids
Acid/base anhydrides
Precipitation reactions
Oxidation-reduction reactions

Figure 9: Sample process identification and abstraction

Decomposition and abstraction

To obtain generic, primitive PSMs from such domain-specific processes, we followed a *divide and conquer* approach, recursively splitting them into their sub-processes until atomic, domain-specific processes were obtained. This is the case of, e.g., the process of a *precipitation* reaction and its two subprocesses *dissolve* and *crystallize.* Atomic processes constitute the basic building-blocks, which can be aggregated to build complex processes.

Atomic processes need to be decontextualized from their original domains in order to guarantee reusability across different domains. We applied the domain-independent terminology of the process metamodel to this collection of domain-specific atomic processes, producing a set of generic, domain-independent process abstractions. The PSM library, containing 15 primitive PSMs, stems from these abstract processes. It establishes and controls the actions required in each of them and defines the necessary knowledge for each process step. The second part of Figure 9 shows two of these PSM (*Decompose* and *Combine*) in the context of a Chemistry *precipitation* process.

4.3. A PSM Library for the Acquisition of Process Knowledge

The two-phased (identification and abstraction) construction of the PSM library produced the domain-independent problem solving methods sketched in Figure 10. The methods contained in the PSM library are primitive methods, i.e., PSMs which cannot be further decomposed into more specific, lower level PSMs. By doing so, PSMs appear to SMEs as simple, self-contained, comprehensible templates for formulating process knowledge, which can be easily aggregated in order to formulate complex processes and edited by SMEs to their convenience while maximizing PSM reusability across the different domains.

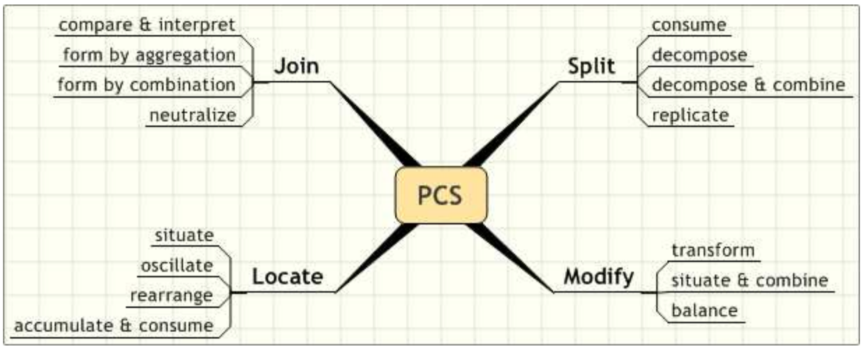


Figure 10: PSM library for process modeling

The PSM library is structured into four main categories (Figure 10), each addressing a different family of processes. *Join* contains PSMs which describe the different ways in which a set of input resources can interact with each other, be combined, aggregated, and, in occasions, neutralize their respective properties, in order to produce a given outcome as a result of such interaction. Other PSMs of this category aim at comparing two different resources by putting them together and study their mutual reaction. *Split,* opposite to *Join*, contains methods that describe several ways of producing a new result by means of consuming or dividing a set of resources into its constituents. *Locate* aims at the spatial arrangement of resources. Finally, *Modify* contains a set of methods that describe different ways of altering either the properties of such resources or the resources themselves, producing a different outcome in the process.

These primitive PSMs define methods to achieve domain-independent, atomic processes. They establish and control the sequence of actions required in each process, but their main goal is to define the necessary knowledge in each process step. The abstractions of these methods allows capturing, in a high-level and generic way, the occurrences of processes in our target domains, formalizing process knowledge by means of mapping their roles against domain entities and aggregating primitive methods for the composition of complex processes.

Next, we describe our PSM library in terms of the process categories identified during our analysis of the syllabus of our target domains and introduce the PSMs[11] that we have designed in order to represent such processes. In the following figures, we show how we have decomposed the main process categories into subcategories and provide a set of PSMs that can be used to support the acquisition of such process types. Graphically speaking, processes are represented as ellipses, while methods are shown as rectangles in a way analogous to the classical task-method notation.

The *Join* category

The *Join category* comprises three different types of processes: *Contrast, Form,* and *Neutralize* (Figure 11). *Contrast* stands for processes where the intended goal is to identify the properties of a given resource by making it interact with another resource whose properties are well known, e.g., a chemical titration. This kind of processes can be achieved by *comparing & interpreting*. *Form* comprises processes where different

[11] A detailed description of all the PSMs contained in the PSM library can be found in APPENDIX A: A PSM Library for Process Knowledge.

resources are put together in order to create a new one. We have produced two different methods that can be applied alternatively in this case (*form by aggregation* and *form by combination*). Finally, while *Neutralize* also stands for processes aiming for a combination of elements, their outcome neutralizes their original properties.

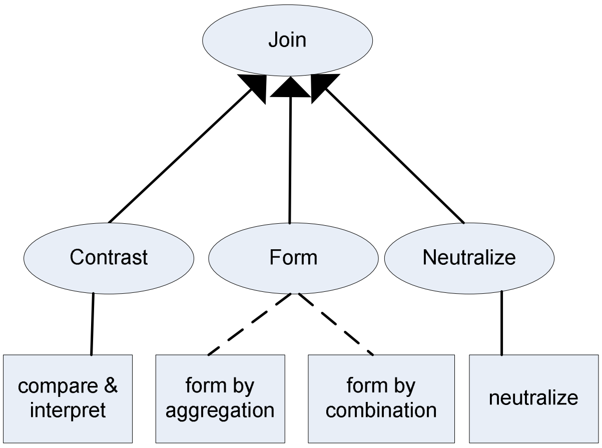


Figure 11: PCS category *Join*

The *Split* category

This category of processes comprises four different process types: *Consume, Decompose, Recombine,* and *Replicate* (Figure 12). *Consume* stands for processes, e.g., a combustion, whose input is spent, providing some byproduct upon termination. *Decompose*, e.g., hydrolysis, divides its inputs into their constituent elements. *Recombine* comprises processes where first decomposition takes place and then, the resulting pieces are combined in order to produce a different output. Examples of *recombine* include sexual reproduction. Finally, *Replicate* appeals to processes where two identical items are produced from a single one, e.g., clonation.

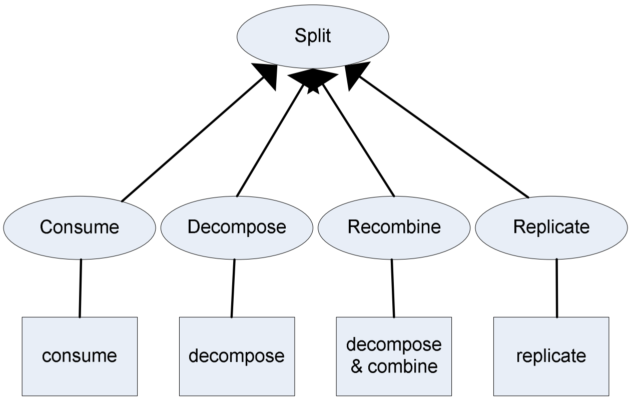


Figure 12: PCS category *Split*

The *Modify* category

This category groups three different types of processes: *Transform, Implement,* and *Balance* (Figure 13). *Transform* stands for processes, e.g., a chemical ionization, whose input is derived into an item of a different type. *Implement*, e.g., implantation, installs an item in a given environment which evolves into something different. Finally, *Balance* appeals to processes where input elements are put together to equilibrate their properties, e.g., osmosis, autoionization of water.

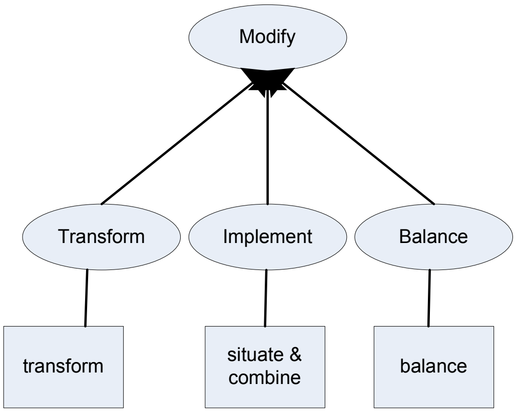


Figure 13: PCS category *Modify*

The *Locate* category

Finally, this process category comprises four different types of processes: *Situate, Oscillate, Rearrange,* and *Release* (Figure 14). *Situate* stands for processes, whose goal is to place their input in a different situation from the original one, understanding situation as a property of such input. *Oscillate* takes this spatial notion into a periodical event, e.g., harmonic motion in Physics. *Rearrange* comprises processes where input items are fetched and arranged in a different disposition, e.g., changes in protein conformation. Finally, *Release* appeals to processes where an agent consumes, upon a given precondition, some items, e.g., hormone secretion.

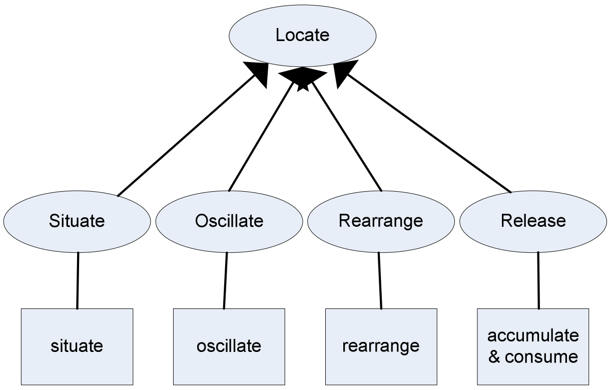


Figure 14: PCS category *Locate*

Many processes in the syllabus are simple processes that occur in the context of larger processes, like, e.g., mitosis, meiosis, and their respective phases. In such cases, primitive PSMs from the PSM library, which can describe the simpler processes (in terms of the succession of steps necessary to accomplish them and the type of knowledge required at each step), can be aggregated to describe the larger processes. On the other hand, the level of refinement to which a process can be represented depends largely on the design approach taken by the SME. For instance, a mitosis process could be represented by applying the PSM *decompose & recombine* from the *Split* category, but that alone would not provide much information on the phases of mitosis (*prophase, prometaphase, metaphase, anaphase,* and *telophase*), each of them a (sub)process itself.

If we look at the syllabus selected for the target domains, the most relevant PSM in terms of appearances in processes described in such syllabus is *transform*, followed by *replicate* and *decompose*. In a nutshell, these three PSM alone from the fifteen available in the PSM library suffice to model 46% of all the processes in the syllabus. The distribution of all the methods of the PSM library in terms of their suitability to formulate and reason with the processes present in the syllabus can be seen in Figure 15.

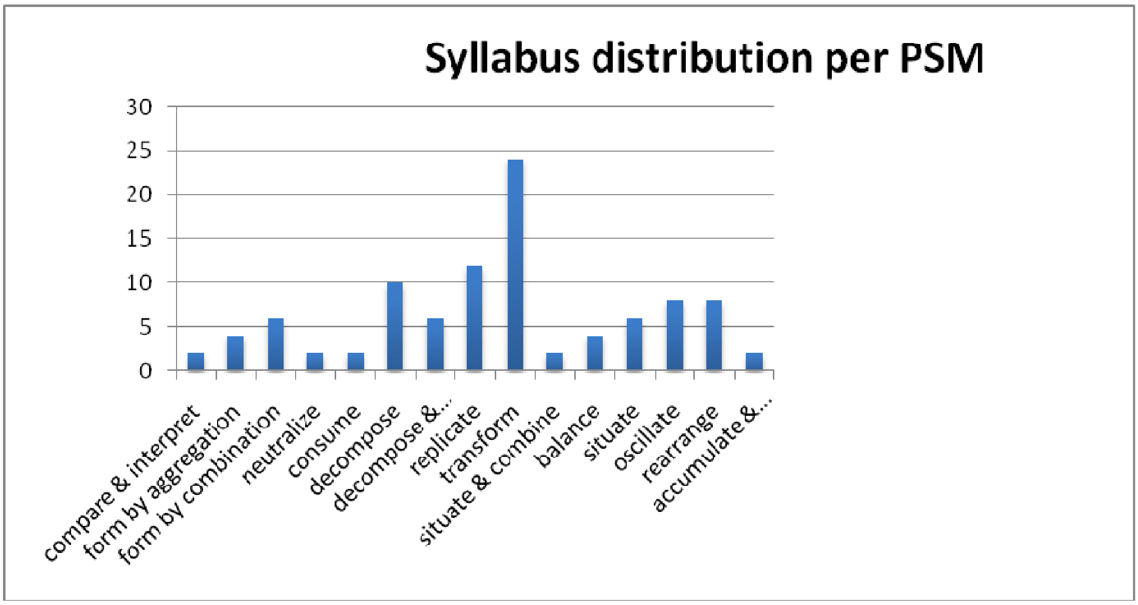


Figure 15: Distribution of the process syllabus across PSM library methods

5. ENABLING SMEs TO FORMULATE PROCESS KNOWLEDGE

One of the main difficulties in knowledge formulation is the gap between domain knowledge and the expertise required in order to formalize and exploit such knowledge. Based on the process metamodel and the PSM library described in this article, it is possible to provide SMEs with a knowledge-level (Newell, 1982) strategy for formalization and reasoning with processes, which enables the creation of process models.

To this purpose, SMEs need usable graphic tools and editors that simplify the manipulation of process knowledge, allowing them to approach the knowledge representation problem exclusively at the domain level. On the other hand, reasoning needs to be supported by means of the automatic synthesis, from user-tailored process diagrams, of executable process models that can be seamlessly integrated with the remaining knowledge types, aiming for a single entry point for question answering.

The DarkMatter *process* perspective (Figure 16) implements this approach, allowing SMEs to formulate domain-specific processes. The process perspective leverages the process metamodel, which provides the required semantics in the form of process terminology, and the PSM library. The PSM library provides SMEs with guidance to model, without the intervention of KEs, well-formed meaningful process diagrams with respect to the underlying process representation and reasoning formalism.[12] The methods of the PSM library can be used by SMEs as knowledge templates, which facilitate building complex processes and alleviate the blank page syndrome. The

[12] The underlying formalism for process knowledge representation and reasoning is described in detail in (Gómez-Pérez, 2009), chapter 5: *Representing and Reasoning with SME-authored Process Knowledge*.

resulting diagrams are automatically encoded, following the formalism, and eventually executed.

The process perspective allows graphically formulating process knowledge either from existing diagrams or from scratch. It provides SMEs with a palette containing all the process entities described in the process metamodel and the methods of the PSM library, allowing Drag & Drop of these elements into the drawing area. Figure 16 shows an example process diagram corresponding to a chemical precipitation as described in (Brown et al., 2002). The precipitation process comprises two different steps: first, the different ionic compounds of a solution are dissolved, then, some of their anions and cations crystallize as a new ionic compound whenever the necessary conditions in terms of temperature and solubility of the ions hold.

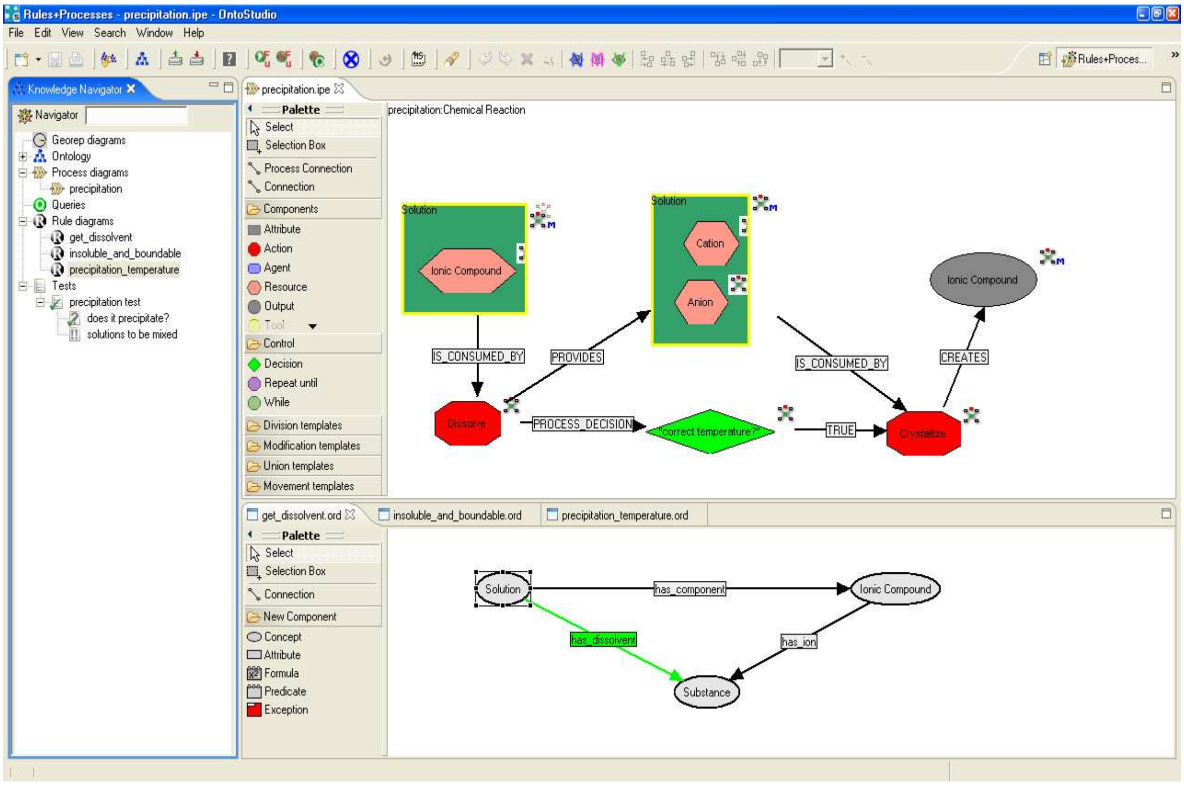


Figure 16: Process Modeling in DarkMatter

Modeling processes in the process perspective is an iterative task that typically involves the following three main stages:

1. **Including the basic process components and structure**, starting with subprocesses. In the example, the precipitation process comprises two subprocesses: *Dissolve* and *Crystallize*. The basic structure of the process model has been provided by the PSM *decompose & combine*, especially suited to model recombination-intensive processes, as defined by the process category *Split* (see Figure 12). PSMs are available from the palette, following the categorization described in section 4, with the twofold goal of simplifying reuse by SMEs and of providing process templates from which to start the modeling task. Though the process perspective supports modeling processes from scratch, using PSMs as templates simplifies the task. The process perspective allows adapting the template, removing and adding process components as required.

2. **Grounding process components in the domain**. During process formulation, generic roles coming either from PSM templates or from the palette as individual components need to be mapped against domain entities in order to contextualize the process in the application domain. Such mappings use the domain-level knowledge base as a bridge between the process knowledge type and the other knowledge types, fundamentally rules, ensuring their convergence in terms of the domain and allowing seamless knowledge representation and reasoning between the knowledge modules.

   The components of a process diagram are mapped against concrete domain entities by means of the interface shown in Figure 17. The figure shows how the domain-level entity *Ionic Compound* is modeled as the process metamodel entity *resource*. In Figure 16, the input to *Dissolve* comprises a *Solution* of *Ionic Compounds,* modeled as a *bag* and *resources* from the process metamodel, respectively. *Solution* is also part of the output of *Dissolve* and of the input of *Crystallize*, together with the *Cations* and *Anions* resulting from the previous subprocess, both modeled as *resources*. The overall output of the process, modeled as a *process output* from the metamodel, is mapped against the domain entity class *Ionic Compound*.

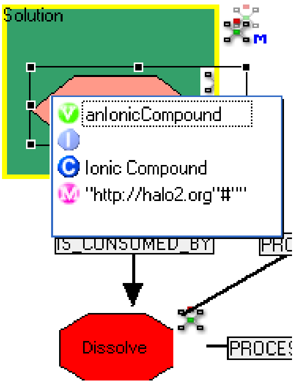


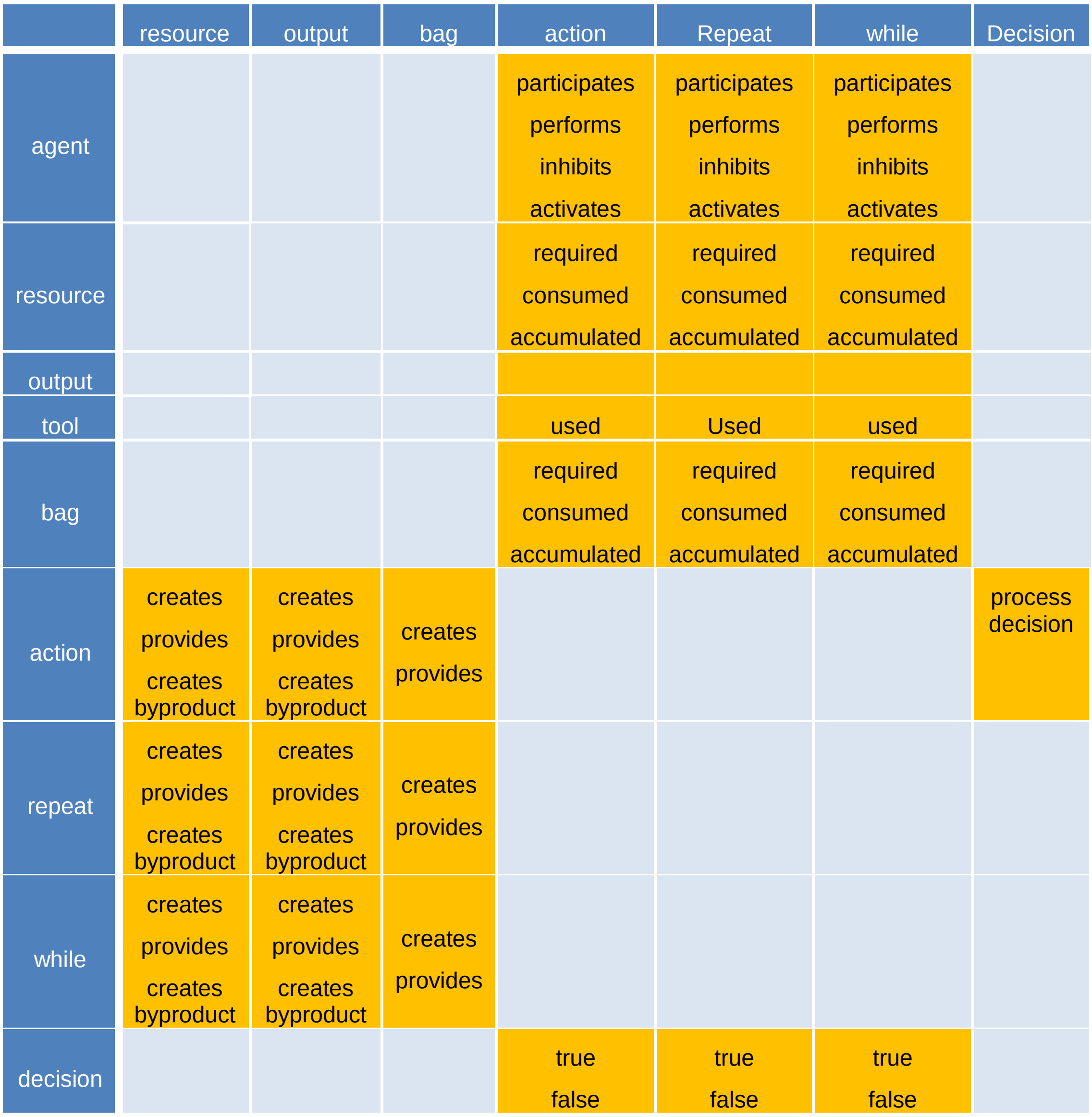
Figure 17: Mapping between role and domain entity

3. **Mapping process components against domain entities**. Finally, the different process components are connected with each other by means of process relations provided by the process metamodel. The process metamodel allows constraining the relations between whatever pair of entities from the metamodel, preventing SMEs from establishing meaningless and confusing links between them. Table 1 shows the allowed relations between the main process entities. When the SME establishes a relation between two process entities, the process perspective automatically retrieves this information from the metamodel and generates a menu that only contains the valid ones.

The process perspective provides an answer to two important challenges in order to enable SMEs to author correct process models

- Ensuring that the resulting process models are compliant with the underlying formalism for process representation and reasoning.
- Guaranteeing that the resulting process models satisfy the modeling expectations of the SME under a functional point of view, i.e., the resulting model must describe the process addressed by the SME and no other.

With respect to the first challenge, the existence of a formalism for process knowledge representation and reasoning explicitly describing the operationalization of knowledge-level process models into a concrete formal language allows dealing with possible inconsistencies introduced by SMEs upon process modeling and facilitates the creation of semi-automatic supervision mechanisms that notify SMEs on modeling mistakes. Such mechanisms contribute to the creation of consistent process models that can be automatically translated into correct code in the reference formal language.



	resource	output	bag	action	Repeat	while	Decision
agent				participates performs inhibits activates	participates performs inhibits activates	participates performs inhibits activates	
resource				required consumed accumulated	required consumed accumulated	required consumed accumulated	
output							
tool				used	Used	used	
bag				required consumed accumulated	required consumed accumulated	required consumed accumulated	
action	creates provides creates byproduct	creates provides creates byproduct	creates provides				process decision
repeat	creates provides creates byproduct	creates provides creates byproduct	creates provides				
while	creates provides creates byproduct	creates provides creates byproduct	creates provides				
decision				true false	true false	true false	

Table 1: Allowed relations between process entities

Additionally, stemming from the above mentioned process representation formalism, the process perspective performs consistency checks of the process diagrams at modeling time, observing a twofold goal: i) to ensure consistency of the process model with respect to the knowledge base and ii) to ensure data flow consistency of the resulting process model.

Process knowledge lies at a higher level of abstraction than other knowledge types like, e.g., classes, instances, and rules. Therefore, a critical point for an effective use of process models is the grounding of the process-level constructs provided by the process metamodel and the PSM library into specific elements of the knowledge base whose process knowledge is being described by such models. Maintaining the consistency of process models in the context of their corresponding domains and knowledge bases contributes to support seamless reasoning at the domain level. In this regard, with respect to the first goal (to ensure consistency of the process model with respect to the knowledge base), the following assumptions are verified:

- Process diagrams must be bound to concrete occurrences of processes from the knowledge base. In the example shown in Figure 16, the process diagram is bound to *precipitation*, an instance of class *Chemical Reaction* in the Chemistry knowledge base.

- The resources used in the process diagrams must be mapped against specific knowledge base entities, either classes or specific instances. In the example, the generic process metamodel entities *bag* and *resource* are thus ground as domain-specific classes *Solution* and *Ionic Compound*, respectively.
- All the relations between process entities must be specified in the process diagram, forming a connected graph, with meaningful nodes and edges.

The assumptions concerning the second goal (to ensure data flow consistency of the resulting process model) are the following:

- Process actions must have inputs and outputs, e.g., *Solution*, *Cation*, and *Anion* (inputs of action *Crystallize* in the example of Figure 16) and *Ionic Compund* (output of the same action).
- Process diagrams must have inputs and outputs, i.e., all processes must be triggered by the occurrence of some entity and produce new knowledge as a result. In the example, a *Solution* of *Ionic Compounds* is the input of the *precipitation* process, which, given the right conditions holds, produces a new *Ionic Compound* as a result.
- Process diagrams must be structured as directed and fully connected graphs, ensuring a consistent data flow.
- The inputs and outputs of process actions must be explicitly represented, ensuring the visual consistency of the action with respect to its semantics. As a consequence, elements occurring both as inputs and outputs of an action are represented as graph nodes before and after the action node. In the example, *Solution*, modeled as a process resource of type *bag*, illustrates this in the process action *Dissolve*.

As a consequence of automatically checking these assumptions, code is produced, in the underlying process representation language, exclusively for well-formed process models. Our modeling-time verification approach supports the detection of data and control flow errors in the process model that fail to fulfill the assumptions, preventing the generation of incorrect code. This method is extremely cheaper, in computational terms, than auditing the code once this is actually generated and hence makes it unnecessary to perform additional checks at the level of the underlying knowledge representation language. When any of these assumptions is violated, the process editor issues an error message explaining the rationale behind it and indicates the SME how to address it.

The actual code corresponding to the process model is actually stored in the knowledge base only when all the assumptions are satisfied. In addition to formulating correct process models, by providing SMEs with this kind of feedback at modeling time we ensure that the resulting process models fulfill the constraints necessary to optimize the generated code as shown in (Gómez-Pérez, 2009).

The second challenge in terms of process correctness (guaranteeing that the resulting process model satisfies the expectations of the authoring SME under a functional point of view) requires enabling SMEs to test the process models. SMEs validate that their process models actually behave as they expect by means of the *test & debug* perspective.

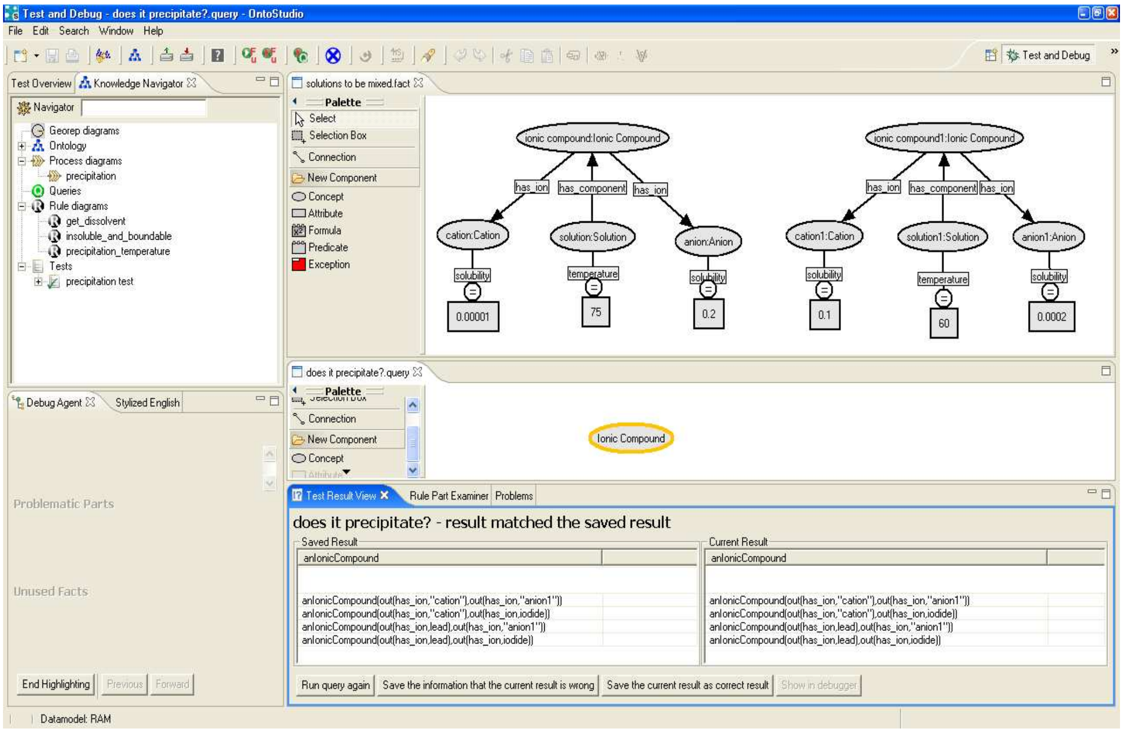
This perspective enables SMEs to create unitary tests that ensure the quality of their knowledge bases remains in good conditions as they modify them. Such tests are especially useful for process modeling, since their inference capabilities can have a substantial effect on the knowledge base. SMEs are therefore encouraged to include a battery of tests (a test set) associated to each process model. A test set consists of a number of queries and optionally a facts file, which contains temporary facts that only live within the scope of the test, i.e., a number of instances for use exclusively within the test set queries. When a test set is created and validated, the results of its execution are saved as a snapshot of the knowledge base. Subsequent executions of the test are regarded as valid if their results match the saved results.

Figure 18 shows a test for the precipitation process. Two solutions, lead nitrate ($Pb(NO_3)_2$) and potassium iodide (*KI*), inform the temporary facts of the fact file for a query that aims at retrieving the final output of the process, i.e., which ionic compounds are produced as a result of a precipitation process. The results returned (lower, right part of the figure) are the combination of the anions and cat-ions that observe the necessary insolubility conditions, i.e., lead iodide (PbI_2) in this case.

Our approach also supports answering meta-level questions on processes, about, e.g., their structure and intermediate states and products. The following illustrates such type of questions:

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



does it precipitate? - result matched the saved result

Saved Result

```
anIonicCompound

anIonicCompound(out(has_ion,"cation"),out(has_ion,"anion1"))
anIonicCompound(out(has_ion,"cation"),out(has_ion,iodide))
anIonicCompound(out(has_ion,lead),out(has_ion,"anion1"))
anIonicCompound(out(has_ion,lead),out(has_ion,iodide))
```

Current Result

```
anIonicCompound

anIonicCompound(out(has_ion,"cation"),out(has_ion,"anion1"))
anIonicCompound(out(has_ion,"cation"),out(has_ion,iodide))
anIonicCompound(out(has_ion,lead),out(has_ion,"anion1"))
anIonicCompound(out(has_ion,lead),out(has_ion,iodide))
```

Figure 18: Process Validation

6. EVALUATION

The process editor and the whole approach towards process knowledge acquisition by SMEs were evaluated by an independent team[13] in the context of the evaluation of the DarkMatter system in Project Halo. A total of six knowledge formulation (KF) SMEs participated, who formulated knowledge on the selected evaluation syllabi for the domains of Chemistry, Biology, and Physics, and tested reasoning with it. These knowledge bases were later used by five Question Formulation (QF) SMEs, with the support of QF KEs, who formulated selected AP-level questions that were intended to be answered by the system. After receiving a limited amount of training on the principles of the system, SMEs used DarkMatter to formulate the knowledge contained in the syllabi. During KF, SMEs were kept isolated from developers, evaluators, and other SMEs to ensure the validity of the evaluation process. In case assistance was necessary, SMEs could ask for support via a chat mechanism.

The overall goal of this evaluation was i) to measure the coverage provided by the solutions proposed to the issues detected in the analysis and design phases and ii) to provide feedback to the development team for tuning and improving the system. Therefore, the scope of this evaluation goes beyond usability aspects in a formative sense, and provides an empirical assessment of the system's coverage and performance in a setting that is representative in terms of the profile of the recruited SMEs and their assigned tasks. In the particular case of process knowledge, since our approach is focused on enabling SMEs to model executable processes at the knowledge level without intervention of KEs, the evaluation paid special attention to direct feedback from SMEs on process knowledge formulation.

In addition to the knowledge bases produced by SMEs, we collected their impressions on the process component according to two main dimensions: usability and utility. The process knowledge bases produced by SMEs during evaluation were checked using the *Test & Debug* perspective through test sets developed by SMEs themselves.

6.1. Evaluation Syllabus

Selecting a syllabus helps specifying any pre-requisite knowledge that must be present in the system before the SME starts to do knowledge formulation. With this syllabus, we aimed to pick material that would definitely pose a challenge for our knowledge formulation systems. Next we describe and justify the selected syllabi for the three target domains.

Chemistry

The following syllabus was selected from the Chemistry reference textbook (Brown et al., 2002), which is representative of the kind of material found in a Chemistry textbook and gives a natural baseline for comparison to the knowledge formation undertaken by KEs.

- Sections 3 3.1-3.2, Pages 75-83 Stoichiometry: Calculations with Chemical Formulae.
- Sections 4 4.1-4.4, pages 113-133, Aqueous Reactions and Solution Stoichiometry .
- Sections 16.1-16.11, pages 613-653 Chemical Equilibrium.

[13] The evaluation team was led by Ergosign GmbH (http://www.ergosign.de).

Additionally, some background knowledge needed to be pump-primed to enable SMEs to formulate the selected syllabus. Sections 3, 4, and 16 require associating chemical names with formulae. Section 16 also required knowing the definition of moles, molarity, equilibrium, and buffer solutions.

Biology

From the reference textbook (Campbell and Reece, 2001), we selected the following syllabus, focused on two main aspects of Biology: the cell and DNA structure and their internal processes. These two main content areas form the basis of much of modern biology and physiology, and are representative of the type of content found throughout the biology domain. From the representation point of view, these subjects deal with central problems in representation of objects and processes. Therefore, they make a good subject matter for evaluation.

- Cell structure and cell processes, including mitosis and meiosis. Pages 112-124, 217-223, and 239-245.
- DNA structure and DNA structure processes, including DNA replication, repair, transcription, and translation. Pages 293-301, 304-311, and 317-319.

Since Biology is a priori one of the most relevant domains for the process knowledge type, before the actual evaluation we had an additional senior Biology SME to analyse the syllabus selected for this particular domain. This biologist used the process metamodel as a framework to formulate process knowledge and produce a Biology ontology on the issues covered by the syllabus. She used the lexicon provided in the metamodel to describe process entities occurring in Biology and, especially, the methods contained in the PSM library as a flexible and reusable mechanism to acquire processes during knowledge formulation.

Physics

The following parts of (Serway and Faughn, 2003) were selected for the Physics domain. This syllabus contains basic knowledge of Kinematics and Dynamics, where, as anticipated by the domain analysis, the expected amount of process knowledge is very scarce.

- Chapter 2: Kinematics - Describing motion in one dimension.
- Chapter 3: Kinematics in two dimensions, except sections 3.1 to 3.4.
- Chapter 4: Dynamics - Newton’s laws of motion.

6.2. Distribution of the Formulated Processes across the Evaluation Syllabus

Table 2 shows the distribution of the processes formulated by the SMEs across the different target domains. The overall number of resulting processes (eleven) is relatively small. The table also shows how processes are usually densely populated with a large number of entities from other knowledge types, like, e.g., rules, for reasoning throughout process steps. The considerably high number of rules used in each process indicates that SMEs succeeded in connecting rule and process knowledge during evaluation.

	# of processes modeled	# of rules imported per process	# total number of rules for all the processes	# average number of rules per process
SME1 (Physics)	0	0	0	0
SME2 (Biology)	2	26 / 2	28	9.33
SME3 (Biology)	6	11 / 7 / 0 / 4 / 8 / 3	33	5.5
SME4 (Chemistry)	0	0	0	0
SME5 (Chemistry)	3	3 / 1 / 1	5	1.66
SME6 (Physics)	0	0	0	0
Total	11	-	66	6

Table 2: Summary of the process knowledge type

Preliminary analyses on Physics anticipated that this domain had a small amount of process knowledge. This has been confirmed by SMEs during evaluation, under the light of the selected syllabus. On the contrary, Biology has the largest knowledge bases and the topics selected for Biology contain large amounts of factual and rule-based knowledge, which is generally true for the domain of Biology.

As expected, we find in Biology the largest population of processes amongst the three domains. SME2 formulated two processes while SME3 formulated six different processes. Both SMEs created tests for their processes with the *test & debug* facilities in order to validate them and to enable reasoning with them in the context of their respective knowledge bases. Most of the PSMs used during the formulation tasks of these Biology processes belong to the categories *Join* and *Split*. *Modify* methods were also used, but to a lower extent.

Finally, Chemistry is at an intermediate point between Physics and Biology in terms of the size of the overall knowledge bases and the number of processes formulated. Chemistry SMEs (specifically SME5) built and validated three different processes using methods from the *Split* and *Join* categories.

6.3. Utilization of the PSM Library and the Process Metamodel

If we describe the different processes modeled by SMEs in terms of the main categories of the process metamodel used upon their formulation (fourth to seventh columns of Table 3), we can draw further conclusions from a different point of view. Process resources and relations are by large the main categories used by our SMEs, with 43% and 42% of the overall process entities, respectively. On the other hand, actions are the third more used category, with 11%. Finally, the use of forks has been merely testimonial (0, 25%).

	Processes	PSMs	# of resources	# of relations	# of forks	# of actions	Total number of process entities
SME2 (Biology)	Transition from G2 phase to mitosis	-	12	21	1	3	**37**
	Mitosis	-	34	37	0	5	**76**
SME3 (Biology)	Mitosis	decompose & combine	29	11	0	5	**45**
	Carbohydrate metabolism	consume, transform	5	6	0	2	**13**
	Cellular respiration	decompose, consume	5	7	0	2	**14**
	Detoxification	transform	4	4	0	1	**9**
	Photosynthesis	form by combination	6	6	0	1	**13**
	Ribosome protein synthesis	situate & combine	2	2	0	1	**5**
SME5 (Chemistry)	Complete ionic equation	form by combination	7	7	0	1	**15**
	Molecular equation	decompose & combine	4	4	0	1	**9**
	Net ionic equation	form by combination	3	3	0	1	**7**
Total			**111**	**108**	**1**	**23**	**243**

Table 3: Occurrences of process metamodel entities

The utilization of the methods contained in the PSM library available in the process editor by SMEs responds to the following pattern. The degree of utilization of the PSMs by SMEs is inversely proportional to the size and complexity of the processes to be modeled. In general, when dealing with particularly complex processes, SMEs need several attempts in order to recognize the advantages of using the available PSMs as knowledge templates that can simplify the process formulation task. On the contrary, SMEs clearly identify relevant PSMs when the processes to be modeled are simpler, probably because it is easier for them to establish a correlation between the process and one single suitable PSM.

This observation is due to two main factors: i) more effort should have been allocated during the training phase to make SMEs more familiar with the PSM library, enabling them to recognize PSMs as a commodity for process formulation and ii) usually SMEs discard PSMs that are suitable for their modeling purposes but need some editing or aggregation work to adapt them to their particular case. That is, though SMEs perceive the value, in terms of reusability and domain independence, of the methods contained in the PSM library, they have problems in perceiving their flexibility.

The third column of Table 3 summarizes the methods from the PSM library that were actually used by SMEs. It is interesting that SME2 did not use any of the available PSMs in order to model either of her two processes, especially if we consider that the overall quality of this SME's knowledge base was substantially lower than the other two. The measure of the quality of each knowledge base was provided by the execution of the test sets created by each SME, using the *test & debug* perspective for the process component. The process knowledge base of SME2 did not successfully pass the corresponding tests. This shows evidence that the generalized use of PSMs for process formulation has provided SME3 and SME5 with means that support building well-formed processes, eventually contributing to higher quality knowledge bases.

Figure 19 shows the distribution of the methods of the PSM library across the modeled processes. This figure, corresponding to the portion of the syllabus addressed by SMEs during evaluation, slightly diverges from Figure 15 (distribution of the process syllabus across the methods of the PSM library), which a priori showed a more relevant role of PSM *transform*, followed by *replicate*, and *decompose*. The characteristics of the subset of the syllabus used for evaluation and the design choices made by SMEs have shifted such relevance towards PSMs like *form by combination*. Additionally, the distribution of the PSMs used is almost uniform across three of the main process categories (*Join*, *Split* and *Modify*).

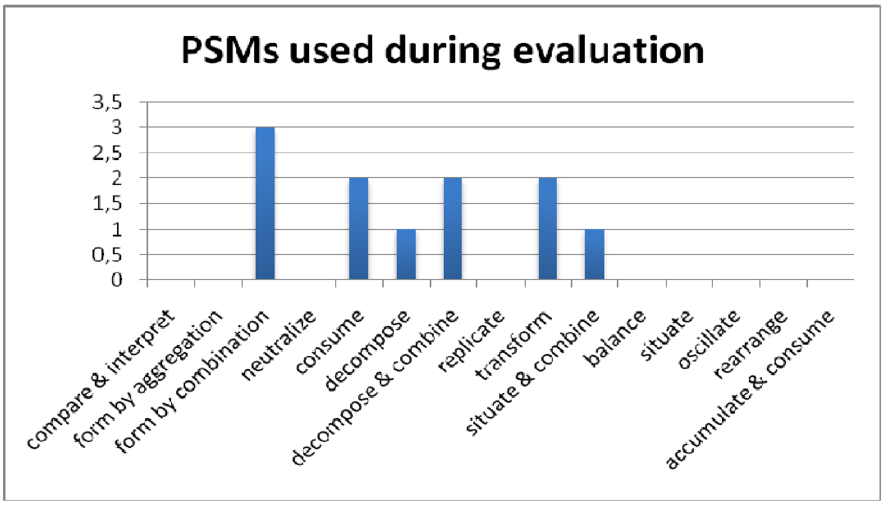


Figure 19: Overall distribution of the PSM library

6.4. Usage Experience of the SMEs with the Process Editor

In general, SMEs only needed some training to get used to the basics of process modeling. However, at the beginning of the evaluation, SMEs did not clearly distinguish the boundaries between the rule and process editors, i.e., what diagram type can be used to represent what kind of knowledge. Additionally, some SMEs tried to use the process editor to assert concepts, instances, or rules, into the corresponding ontology. Table 4 summarizes the issues raised by SMEs during the evaluation, showing the number of questions about the behavior of the different components of DarkMatter.

	Knowledge Navigator	Rule Editor	Mathematica	Processes	Test & Debug	Explanation	WYSIWYM	Help
SME 1	7	13	21	0	6	3	3	0
SME 6	0	0	7	0	2	1	0	0
SME 2	0	3	0	4	7	7	2	0
SME 3	1	12	1	13	5	0	6	0
SME 4	0	10	1	2	6	0	1	0
SME 5	0	1	1	0	2	0	1	0
Total	8	39	31	19	28	11	13	0

Table 4: Issues raised by SMEs about processes in the different domains

After evaluation, we polled SMEs for direct feedback on how much the system helped them to achieve their goals. This is a relevant indicator in a system aimed at knowledge-level process formulation and reasoning like this, which intends to support SMEs to achieve these tasks without intervention of KEs. In this direction, two dimensions have been explored (usability and utility) whose results are detailed next.

Usability

Since DarkMatter components are perceived as a whole, integrated system by the users, usability was measured in an integrated manner for the whole system. In this regard, the System Usability Scale (Brooke, 1996) was adopted and the following questions were posed to SMEs:

- *I would recommend DarkMatter to be used by others.*
- *I found DarkMatter unnecessarily complex.*
- *I think DarkMatter was easy to use.*
- *I think that I would need the support of a technical person to be able to use DarkMatter.*
- *I found that the various functions in DarkMatter were well integrated.*
- *I think there was too much inconsistency within DarkMatter.*
- *I would expect that most people would learn to use DarkMatter very quickly.*
- *I found DarkMatter very cumbersome to use.*
- *I felt very confident using DarkMatter.*
- *I had to learn a lot of things before I could get going with the DarkMatter.*

SMEs were asked to answer each question with a qualitative value ranging from *strongly agree* to *strongly disagree*, which were then transformed into a quantitative value between 1 and 100. Generally, scores around 65 (60-69) reflect average or acceptable satisfaction. Scores below 60 suggest poor satisfaction, and those over 70 suggest good satisfaction.

Table 5 shows the actual usability score as rated by the different SMEs for DarkMatter, including the process component. The average score is 64.5, reflecting an intermediate satisfaction level, acceptable but still improvable. The fact that SMEs using the process component (SME2, SME3, and SME5) provided low scores suggests that the process component needs improvement in terms of usability. This is comprehensible since the process knowledge type is considerably more complex than the remaining knowledge types, and so is the tool itself. However, more intensive training could have probably leveraged this measure.

SME1 (Physics)	62
SME2 (Biology)	50
SME3 (Biology)	62.5
SME4 (Chemistry)	87.5
SME5 (Chemistry)	50
SME6 (Physics)	75

Table 5: SUS scores per SME and domain

Utility

In addition to the previous questionnaire, during the final interviews, SMEs were asked about their impression concerning the utility of the different components for achieving the specified goals by indicating their compliance to the following statement: “I found this component useful in helping me to achieve my goals”. SMEs rated their compliance on a scale from 0 - 4 (*strongly disagree* - *strongly agree*). In order to elicit more detailed information about their attitude to the tool under consideration, the interviewee was also asked about potential improvements that would increase the utility of the component.

Figure 20 shows the utility ratings on the process component. Since the process component was not used for Physics, ratings concentrate on Biology and Chemistry. Chemistry SMEs (SME4 and SME5) did not perceive process as very useful. This can be due to more training being needed in this domain, especially for using the *test & debug* component in order to validate the formulated processes and the resulting overall knowledge bases. On the contrary, in Biology, where the process knowledge type is more evident (and the concentration of process knowledge for the selected syllabus is more representative of the domain than in the Chemistry syllabus), SMEs rated processes as very useful.

Personal interviews with SMEs, especially SME3, who used the process component more intensively, show a high degree of satisfaction with respect to the tool. Specific comments from SMEs are as follows:

- **SME1 (Physics)**: *“I didn’t use it... Had no reason to use it”*.
- **SME6 (Physics)**: *“I didn’t use it at all*”.
- **SME2 (Biology)**: *“It makes the representation of biological models easier”*.
- **SME3 (Biology)**: *“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”*.
- **SME4 (Chemistry)**: *“In Chemistry, possible to set by without it; only in one case necessary: dissolution; with a better reliability, it would be useful”*.
- **SME5 (Chemistry)**: *“I had some trouble coming along, how to use processes for Chemistry... maybe it isn’t that important for Chemistry”*.

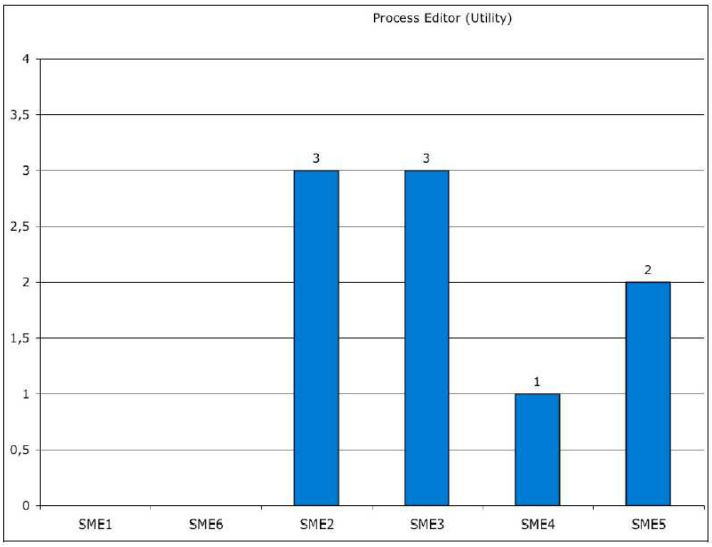


Figure 20: SME-rated utility of processes

7. EVALUATION: SUMMING UP

In the previous section we have described the evaluation of our approach towards the acquisition of process knowledge by SMEs in the context of Project Halo. Now, we connect and contrast the results of the evaluation against each of the expected outcomes presented in the introduction of this article. Since the evaluation was constrained to six SMEs, these results must be analyzed from a qualitative rather than a statistical point of view, as follows:

1. **Higher quality and less costly knowledge bases of processes**, by empowering SMEs and taking KEs out of the process knowledge acquisition loop.

Providing SMEs with the tools and methods needed to enable them to formulate knowledge can significantly reduce costs of acquiring complex knowledge types like processes by taking KEs out of the KA loop. In order to accomplish this vision, it is necessary to abstract SMEs from the difficulties of the underlying representation languages, simplifying their modeling tasks for the process knowledge type.

In the evaluation, the PSM library clearly helped SMEs in their modeling tasks. Furthermore, the SMEs using the methods from the PSM library to acquire process knowledge (SME3 and SME5) produced more and significantly better quality process models than those who did not use them (SME2). The quality of the resulting knowledge bases was determined by the tests created by the SMEs themselves in order to check that their process models actually behaved as expected. 82% of the process models (in general, those whose authors used the methods of the PSM library in a regular basis) were correct. In all cases, the process models were formulated by SMEs without intervention of KEs, who only required initial training and sporadic support in the utilization of the tools.

Therefore, it is possible for SMEs to acquire and formalize process knowledge without the intervention of KEs through the utilization of the methods and tools provided by our approach. As a consequence, this reduces the costs of generating the resulting knowledge bases. Additionally, the evaluation indicates, but does not provide empirical proof, that the resulting SME-authored knowledge bases are at least of such a good quality as those produced by KEs since i) they have been checked by test batteries produced by SMEs themselves and ii) being produced by the SMEs, they reflect directly their expertise of the domain.

2. **Reduced complexity of acquiring process knowledge by SMEs through the use of PSMs** as domain-independent, reusable abstractions of domain-specific processes.

The fact that the rate of correct process models produced by the SMEs who made a more intensive use of the PSM library during knowledge acquisition was by far higher than those who did not use the PSM library indicates that PSMs succeeded in providing SMEs with the required level of abstraction to formulate process knowledge. The utilization of PSMs as domain-independent process templates contributed to reduce the complexity of the modeling task by SMEs, alleviating the blank page syndrome and providing SMEs with modeling guidelines about how to build process models and the type of knowledge required at each process step. Additionally, the approach was welcomed by the SMEs, as shown by their answers to our usability and utility studies.

3. **Keeping acquisition of process knowledge at the knowledge level** through an underlying process knowledge representation formalism transparent from SMEs.

The process component provides SMEs with means to model processes without worrying about the encoding of such process models in a particular language, thus keeping acquisition of processes at the knowledge level. The synthesis of executable code for actual reasoning with the formulated process models is completely transparent from SMEs, who can therefore focus on modeling. As a consequence, SMEs interact with the system in terms of their own domains, avoiding the knowledge acquisition bottleneck.

4. **Flexible and reusable mechanisms for acquisition and reasoning with process knowledge by SMEs**, maximizing the application of the approach across several domains with little effort.

Both the process metamodel and the PSM library have been used in the context of two of the three target domains (Chemistry and Biology), showing evidence of their reusability in this context. As to the graphical process modeling and reasoning environment, it is domain-independent and can be directly applied in any domain. Reusing the whole approach in other domains would however require updating the PSM library as necessary for such domains. The positive experience with the support currently provided for this task encourages the application of our approach to other domains, such as Business or Ecology.

8. DISCUSSION AND FUTURE WORK

Building executable systems from conceptual descriptions is not something new. Some of the most important issues discussed in this article have been already addressed in the literature, including modeling at the knowledge-level, allowing developers to build knowledge systems focusing on logical rather than operational aspects, and the development of structured libraries of reusable PSMs. However, the main difference between these approaches and the work described herein concerns the target users.

While earlier systems aimed to support KEs in building knowledge intensive applications, we focus on creating tools that allow SMEs themselves, without the intervention of KEs, to author scientific knowledge of a particular type: processes.

Process knowledge is one of the most widely used and complex types of knowledge, even beyond the target domains treated herein. Domains like business, healthcare, climate prediction and ecology in general are rich in process knowledge. In order to enable SMEs to model and reason with processes, tools must abstract SMEs from the formal representation and enactment of process-oriented reasoning, based on knowledge entities that can be reused across different domains with a reduced cost. Unfortunately, it is not sufficient to have a process metamodel describing individual process entities, e.g., actions, resources, etc, which must be connected like Lego blocks in order to build potentially large and complex processes, since SMEs lack the required knowledge about how to combine those pieces.

Thus, SMEs need guidance that allows them to build well-formed, consistent process models. Ideally, this guidance could be provided by a process ontology that described the most frequent process types and could be reused across different domains. However, complex, domain-specific processes are difficult to generalize into abstract and reusable process classes, and therefore any process ontology is very unlikely to provide enough coverage to support each of the processes of one or several domains. On the other hand, processes can be decomposed into subprocesses, producing atomic processes that, quite on the contrary, are indeed amenable to be abstracted into (primitive) PSMs as high-level but still easy to manipulate building blocks. This provides SMEs with the required modeling guidance, enabling SMEs to build complex processes by aggregating such domain-independent PSMs as process knowledge templates and instantiating them in their own domains.

Final interviews with SMEs showed that the process component could be improved in terms of usability and also that the training received was not enough in order to completely master its functionalities. Nevertheless, the combined use of the process metamodel and the PSM library allowed SMEs to model all the process knowledge occurrences identified in the evaluation syllabus. In any case, though the use of PSMs provides SMEs with extended modeling capabilities for representing processes, it is still hard for SMEs to realize how to combine these process knowledge templates in order to build complex processes. This problem, derived from the bottom-up nature of our approach, could be addressed by means of an ontology that describes the multiple combination patterns of the different PSMs, taking into account the context provided by the domain of the process being modeled.

Additionally, the evaluation showed that increasing the expressiveness of the process metamodel will be necessary in order to allow representing more complex process constructions from different domains and enabling more sophisticated reasoning that supports answering questions addressing extensive flow characteristics of process knowledge. For example, scientists designing a complex biotechnology experiment will need to reason on the different stages comprised in the process, e.g., regarding certain process steps as optional or context-specific, that need to be expressed by our metamodel. This will help SMEs to understand the consequences of adding a new stage to their processes or what process steps would be necessary and how they should be arranged in order to achieve a certain outcome of the overall process.

Though large part of the PSMs from the PSM library are abstract enough to be reused in domains different from the scientific ones discussed here, a detailed analysis of those domains would probably reveal new abstract processes that would lead to the inclusion of new methods in the PSM library. Furthermore, the same process abstractions can be named differently in different domains. For an appropriate reuse in other domains, SMEs need to be provided with the required lexical resources. An interesting line of work that can simplify the modeling task for SMEs is the use of NLP techniques for the detection of process instances in textual sources and the identification, from amidst the PSM library, of the PSMs suitable for modeling them. We have preliminarily explored this path but, since processes can be described in such various ways using free text, even implicitly, as, e.g., in textbooks, their identification is extremely complicated beyond detection of the most frequently used verbs in processes and the process metamodel entities and their synonyms.

In summary, we have shown that it is possible to engage SMEs in formulating complex knowledge like processes. However, the solutions that we have proposed live on the ground of the *Semantic Web in the Small.* We have adopted relevant Semantic Web principles like ontologies and PSMs and applied them to the formulation of processes by SMEs in a well-defined scenario, with a limited number of SMEs, and well-known, complete, and reliable sources of information. The application of the methods presented in this article to the open and unregulated scenario proposed by the WWW anticipate exciting research challenges related with the acquisition, sharing, and reuse of process knowledge by communities of online users. Such challenges include technical issues, such as nonmonotonicity, dealing with possible inconsistencies of distributed but interacting knowledge bases, as well as trust issues, performance, and scalability.

ACKNOWLEDGMENTS

This work has been funded by Vulcan Inc. as part of the DarkMatter project led by Ontoprise within the Halo program (www.projecthalo.com). This work is also partially supported by the IST-2005-027595 EU project NeOn. We also acknowledge André Valente, for his contribution to the analysis phase of this work, and the ISI team.

REFERENCES

Benjamins, V.R.: Problem Solving Methods for Diagnosis And Their Role in Knowledge Acquisition, *International Journal of Expert Systems: Research and Application*, 8(2):93—120, 1995.

Bock, C. and Grüninger, M. *PSL: A Semantic Domain for Flow Models.* Software and Systems Modeling Journal, vol. 4, pages 209--231, 2005.

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: Taylor and Francis.

Brown, T., Lemay, H., Bursten, B., Burdge, J. “Chemistry: The Central Science”, Prentice Hall, 9th edition, 2002. ISBN: 0130669970.

Campbell, N., Reece, J. “Biology”, Pearson Higher Education, 6th edition, 2001. ISBN: 0805366245.

Crubézy, M. and Musen, M.A. (2003) Ontologies in Support of Problem Solving. In Staab, S. and Studer, R., editor, Handbook on Ontologies in Information Systems, International Handbooks on Information Systems. Springer.

Feigenbaum, E. (1977). The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA.

Fensel, D., Benjamins, V.R., Motta, E. and Wielinga, B.J. UPML: A Framework for Knowledge System Reuse. IJCAI 1999.

Fensel, D., Angele, J., Studer, R. The Knowledge Acquisition and Representation Language KARL. In *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 4, 1998.

Fernández-López M., Gómez-Pérez A., Juristo N. (1997) *METHONTOLOGY: From Ontological Art Towards Ontological Engineering*. Spring Symposium on Ontological Engineering of AAAI. Stanford University, California, pp 33–40.

Friedland, N., Allen, P., Matthews, G., Witbrock, M., Baxter, D., Curtis, J., Shepard, B., Miraglia, P., Angele, J., Staab, S., Moench, E., Oppermann, H., Wenke, D., Israel, D., Chaudhri, V., Porter, B., Barker, K., Fan, J., Chaw, S., Yeh, P., Tecuci, D., & Clark, P. (2004). Project Halo: Towards a digital Aristotle. AI Magazine, 25(4), 29−48.

Gómez-Pérez, J.M., Erdmann, M., Greaves, M. *Applying Problem Solving Methods for Process Knowledge Acquisition, Representation, and Reasoning*. 4th International Conference on Knowledge Capture (KCAP), 2007, Whistler, Canada.

Gómez-Pérez, J.M. *Acquisition and Understanding of Process Knowledge Using Problem Solving Methods.* PhD thesis, Universidad Politécnica de Madrid, Madrid, 2009.

Kifer, M., Lausen, G., Wu, J. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42, 1995.

Marcus, S., Stout, J., McDermott, J. VT: An Expert Elevator Designer that Uses Knowledge-Directed Backtracking. *AI Magazine*, 9(1):95-112, 1998.

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.

Morley, D. and Myers, K. The SPARK Agent Framework, in Proc. of the Third Int. Joint Conf. on Autonomous Agents and Multi Agent Systems (AAMAS-04), New York. NY, pp. 712-719, July 2004.

Motta, E. (1999). Reusable Components for Knowledge Modeling. IOS Press, Amsterdam. November 1999.

Motta, E. (1998). An Overview of the OCML Modeling Language, The 8th Workshop on Knowledge Engineering Methods and Languages (KMEL' 98).

Newell, A. "The Knowledge Level". Artificial Intelligence, 18(1):87-127, 1982.

Ohno-Machado, L., Gennari, J., Murphy, S., Jain, N., Tu, S., Oliver, D., Pattison-Gordon, E. "The GuideLine Interchange Format: A Model for Representing Guidelines", *Journal of the American Medical Informatics Association,* 5(4):357-372, 1998.

Preece, J., Rogers, Y., Sharp, H. Interaction Design: Beyond Human-Computer Interaction. John Wiley and Sons, New York, 2002.

Punch, W. (1989). *A diagnosis system using a task integrated problem solver architecture (TIPS), including causal reasoning.* PhD Thesis, Ohio State University.

Pylyshyn, Z.W. The Robot's Dilemma: The Frame Problem in Artificial Intelligence. Norwood, 1987.

Schreiber, A.Th., Akkermans, J., Anjewierden, A., De Hoog, R., Shadbolt, N., Van De Velde, W., and Wielinga, B.J. *Knowledge Engineering and Management: The CommonKADS Methodology* (2000). MIT Press. ISBN 0-262-19300-0.

Serway, R., Faughn, J. "College Physics", Brooks Cole, 6th edition, 2003. ISBN: 0534492592.

Uschold, M., King, M., Morales, S., Zorgios, Y. "The enterprise ontology", Knowledge Engineering Review, 13(1):31—89, 1998.

Valente, A. and Team Omniscience. Project Halo Analysis Report, May 2004.

Wielinga, B.J., Schreiber, A. T., Breuker, J. A. "KADS: A modeling approach to knowledge engineering" Knowledge Acquisiton, 4(1):5-53. Special issue 'The KADS approach to knowledge engineering'. Reprinted in: Buchanan, B., Wilkins, D. editors (1992), Readings in Knowledge Acquisition and Learning, San Mateo, California, Morgan Kauffmann, pp. 92-116.

APPENDIX A: A PSM Library for Process Knowledge

This appendix contains an extensive description of the PSMs contained in our PSM library. The graphic representation chosen to describe the methods is the PSM knowledge flow view. This perspective supports SMEs during knowledge acquisition, bridging the gap between the particular, domain-specific process being formulated by SMEs and the generic, domain-independent PSMs, whose roles are grounded into the corresponding domain for that particular process.

For each particular PSM, occurrences of the higher-level definition of resources (roles), decisions, and actions, as in Figure 4, Figure 5, and Figure 6, respectively, are extended with more specific terms from our process metamodel, which refine them. Additionally, the description of each PSM is accompanied by a table that summarizes its properties[14].

Category *Join*

Form processes

The PSM library contains two different methods which can be applied to achieve a process of the *Form* type: *form by combination* (Figure 21) and *form by aggregation* (Figure 22). The output of a combination method is a completely different element from the input, while in the case of an aggregation the input does not lose its properties. Method *form by combination* can be applied to processes like, e.g., a chemical precipitation or an ionic binding. In Biology, this method can be applied to processes like, e.g., fusion (a firm association between two cellular components). On the other hand, *form by aggregation* can achieve processes like, e.g., chemical mixes or biological attachment processes. In comparison to fusion, in these processes a loose and temporally restricted association between substances occurs (e.g., an attachment of a tRNA molecule to an mRNA codon).

name	form by combination
Goal	$\exists$ ep, cp member(combination set, element) and property(element, ep) and property(combination, cp) and ep $\neq$ cp
input roles	combination set, combinatory
output roles	combination, byproduct

[14] For a more simple notation, we assume variables (representing method roles) to be universally quantified in the goal expression. Additionally, predicates member/2, property/2, interpretation_of/3, is_byproduct_of/2, part_of/2, interact/2, and layout/2 contribute to simplify the goal expression.

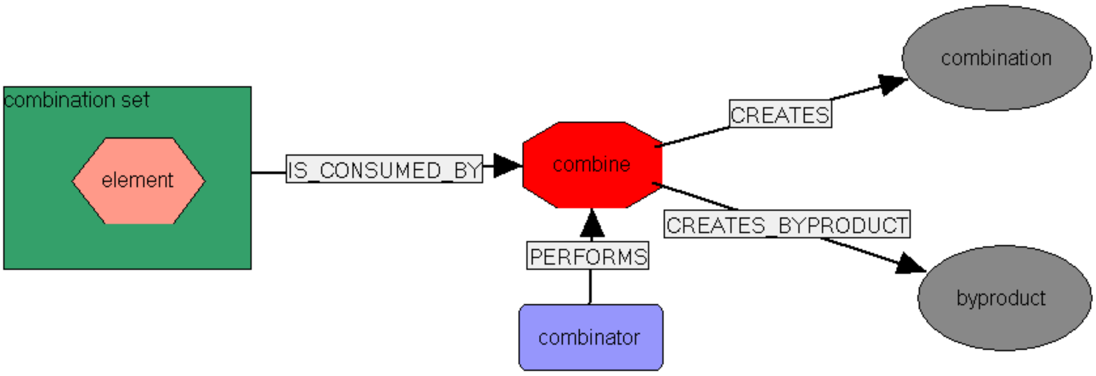


Figure 21: PSM *form by combination*

Name	form by aggregation
Goal	member(aggregation set, element) and member(aggregation, element)
input roles	aggregation set, aggregator
output roles	Aggregation

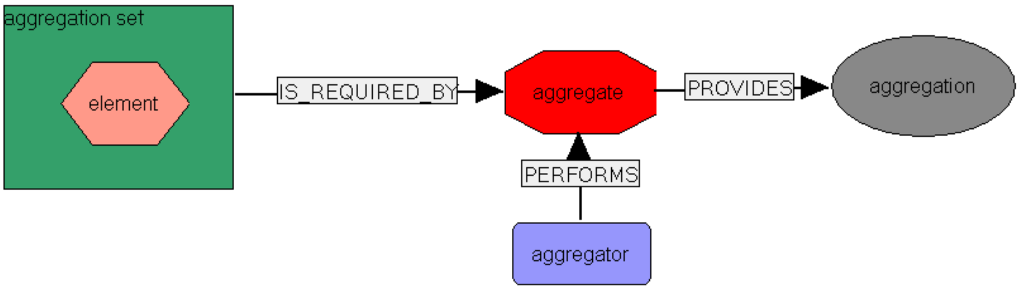


Figure 22: PSM *form by aggregation*

Contrast processes

Process of putting together an item with well-known property/ies and another one whose property/ies is/are totally or partially ignored, and draw a series of conclusions from the results of such interaction. PSM *compare & interpret* can be applied to this process category by dividing the original process in two subprocesses: *compare* and *interpret*. During the first subprocesses, the reference element is compared against the element whose properties are unknown. Then, the observations obtained are interpreted and a conclusion, i.e., the contrast, is produced.

name	compare & interpret
goal	interpretation_of(pattern element, element, contrast)
subactions	compare, interpret
input action	Compare
output action	Interpret
input roles	pattern element, element
output roles	Contrast

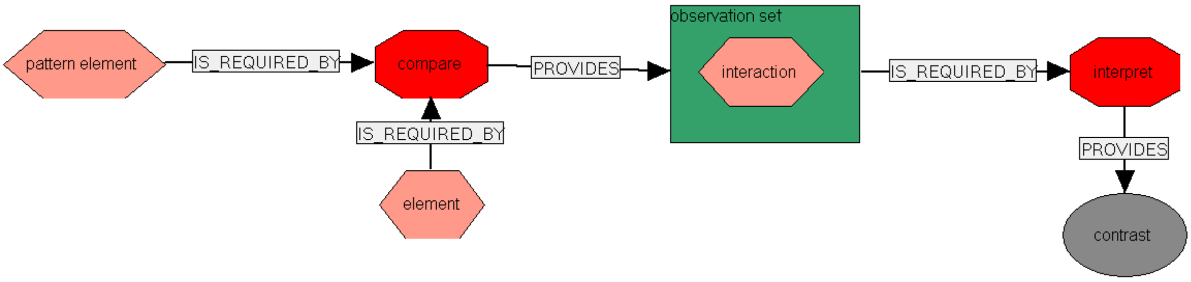


Figure 23: PSM *compare & interpret*

Titration, in Chemistry, is an example of a contrast process where a solution of unknown concentration (role *element* in PSM *compare & interpret*) reacts with one of known concentration (*pattern element*). The point at which stoichiometrically equivalent quantities are brought together is known as the equivalence point of the titration and determines the concentration of the unknown solution (*contrast*). Titrations can be conducted using acid-base, precipitation, or redox reactions.

Neutralize processes

PSM *neutralize* specializes *form by combination* to describe processes whose output is the result of combining its input in a way that the relevant properties of those element are no longer present in the output.

name	neutralize
goal	member(neutralization set, element) and property(element, ep) and property(neutralization, cp) and $ep \cap cp = \emptyset$
input roles	neutralization set, neutralizer
output roles	neutralization, byproduct

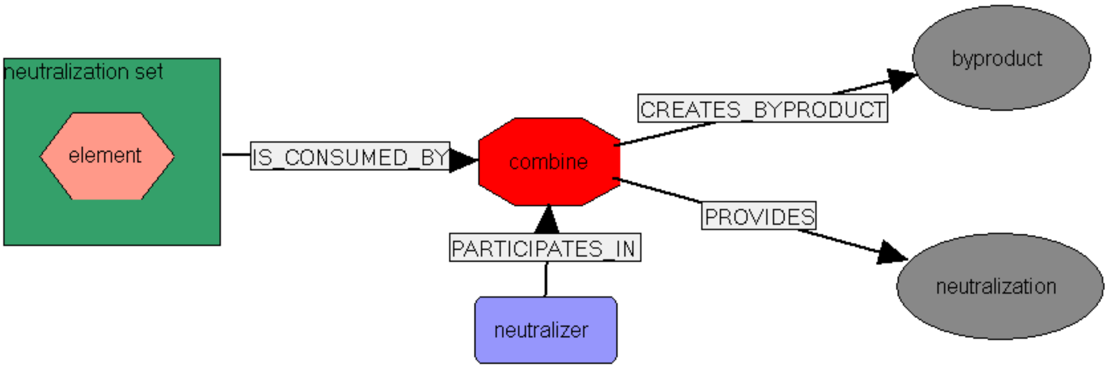


Figure 24: PSM *neutralize*

An example of neutralization are chemical reactions between the *neutralization set* formed by an acid (*element)* and a base (*element) in* order to produce a salt (*neutralization*) and water (*byproduct*).

Category *Split*

Consume processes

Items undergoing this process do not exist anymore. Upon termination, neither any so called product has been created, only byproducts.

name	consume
goal	member(consumable set, element) and is_byproduct_of (waste, element)
input roles	consumable set, consumer
output roles	byproduct

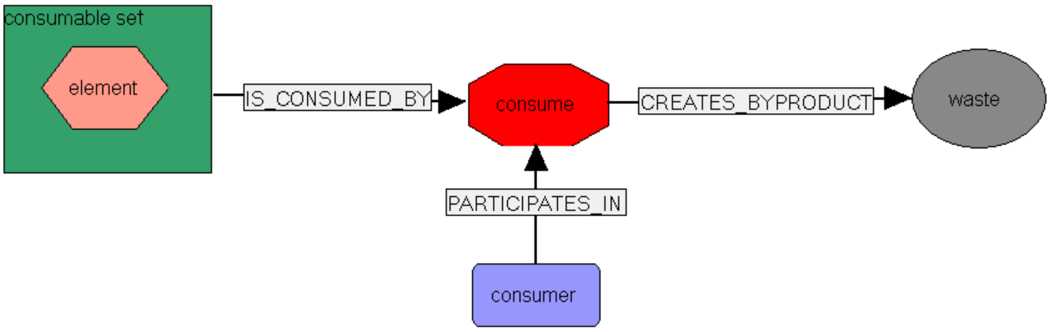


Figure 25: PSM *consume*

Combustion is an example of consumption where the combustible (*element*) is burnt by fire (*consumer*) to produce byproducts like CO2 and H2O (*waste*)

Decompose processes

PSM *decompose* is the reverse from *form by aggregation*. Processes achieved by this PSM take an element and produce two or more other items upon termination.

name	decompose
goal	member(constituent set, piece) and part_of(piece, element)
input roles	element, decomposer
output roles	constituent set

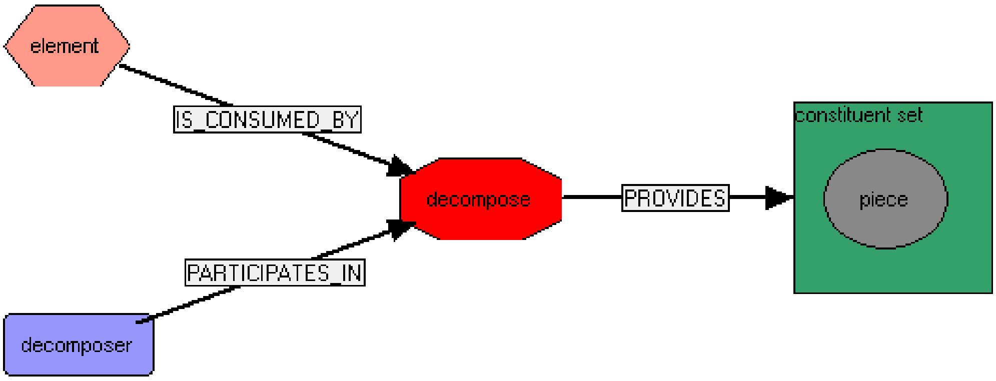


Figure 26: PSM decompose

Examples of decomposition processes are distillation, filtration, and photodissociation. Other examples are hydrolysis, a chemical process where a substance is split in parts by the addition of water molecules, and the formation of the cleavage furrow. The cleavage furrow is a structure developing during mitosis. The middle of the dividing cell grows inward until the two new cells are separated.

Recombine processes

PSM *decompose & combine* can be applied to this process category by dividing the original process in two subprocesses: decompose and *combine*. During the first subprocesses, the input is divided into its constituent elements, which are then put together to produce te overall output. It is a specialization of PSM *form by combination.*

name	decompose & combine
goal	member(recombination set, element) and member(constituents set, piece) and part_of(piece, element) and part_of(piece, combination) and ∃ ep, cp property(element, ep) and property(combination, cp) and $ep \neq cp$
subactions	decompose, combine
input action	decompose
output action	combine
input roles	recombination set, decomposer, combinatory
output roles	combination, byproduct

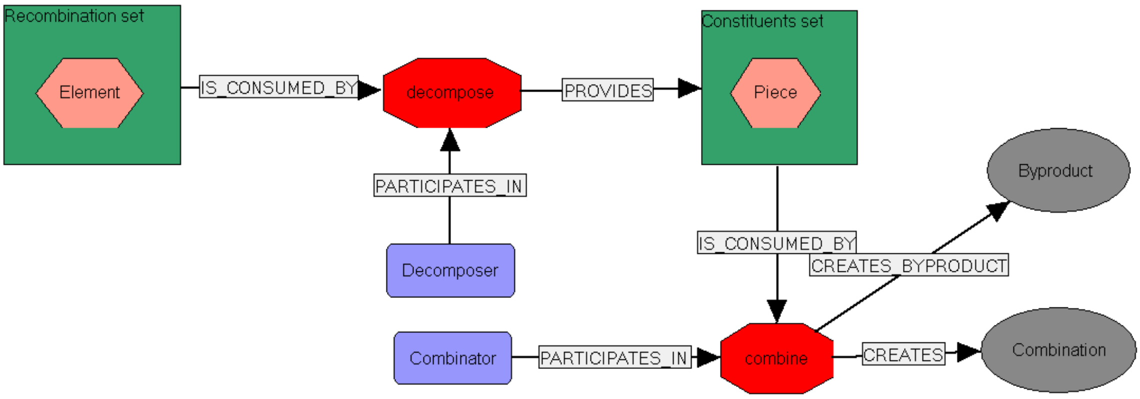


Figure 27: PSM d*ecompose & combine*

Examples of application of this PSM include reformation, where cellular components which have been destroyed (in part) during some process can afterwards be restored to build new structures, and digestion. Digestion splits nutrients into its finest components, which can then be used to build new cellular tissues. This PSM has also been used to formulate the precipitation process in Figure 16.

Replicate processes

PSM *replicate*, achieving this category of processes, copies its input and produces an exact replica.

name	replicate
goal	element = replica
input roles	element
output roles	element, replica

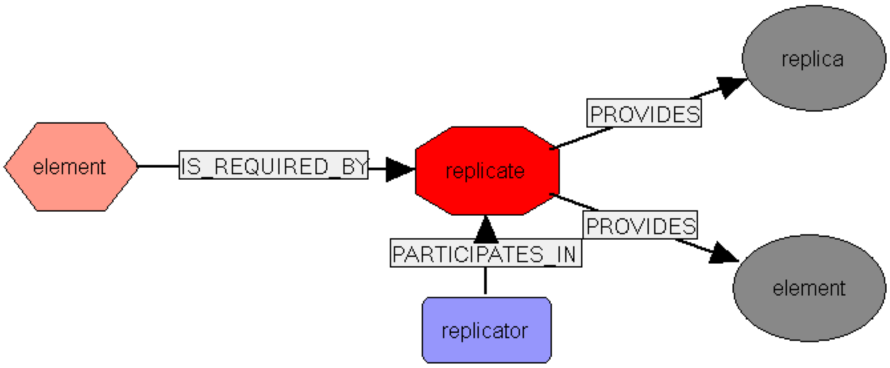


Figure 28: PSM *replicate*

Biology seems to be the domain where a large number of replicative processes occur, e.g., those related to the cell cycle, DNA replication, sexual reproduction, transcription, translation, and binary fission.

Category *Modify*

Transform processes

name	transform
goal	transformed element ≠ element
input roles	element, adapter
output roles	transformed element

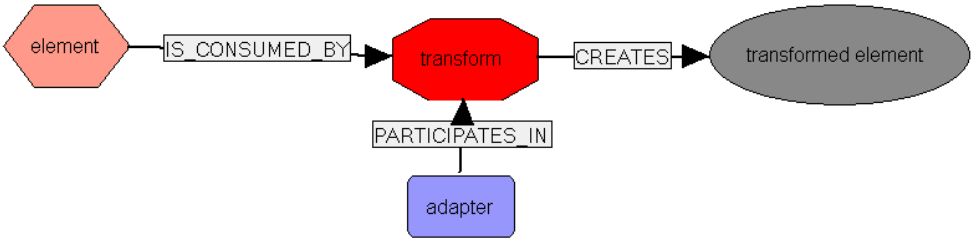


Figure 29: PSM *transform*

Examples of transformation processes are cell growth (a change in size and structure of a cell), changes in chemical structures, e.g., posttranslational modification, RNA-processing, and metabolic reactions, synthesis processes during which some cellular component or substance is produced, e.g., protein, DNA, RNA, membrane, or organelles, detoxifications, when accompanied by a change in the chemical structure and function, and energy transformations, e.g., cellular respiration and photosynthesis, where the energy of sunlight is transformed into chemical energy.

Implement processes

PSM *situate & combine* can be applied to this process category by dividing the original process in two subprocesses: *situate* and *combine*. During the first subprocess, the new element is settled in its new environment. Then, both combine in order to produce an evolution of the later.

name	situate & combine
goal	∃ ep, cp, vp property(element, ep) and property(environment, vp) and property(combination, cp) and ep ≠ cp and vp ≠ cp
subactions	situate, combine
input action	situate
output action	combine
input roles	element, environment, driver, combinator
output roles	combination

Embryogenesis is an example of process that can be achieved by this PSM. It is the process of cell division and cellular differentiation which leads to the development of an embryo, occurring in both animal and plant development. Mitosis happens all through the process, generating more and more cells, making the embryo grow. The blastocyt grows and invades the uterus where it stays until the end of its development.

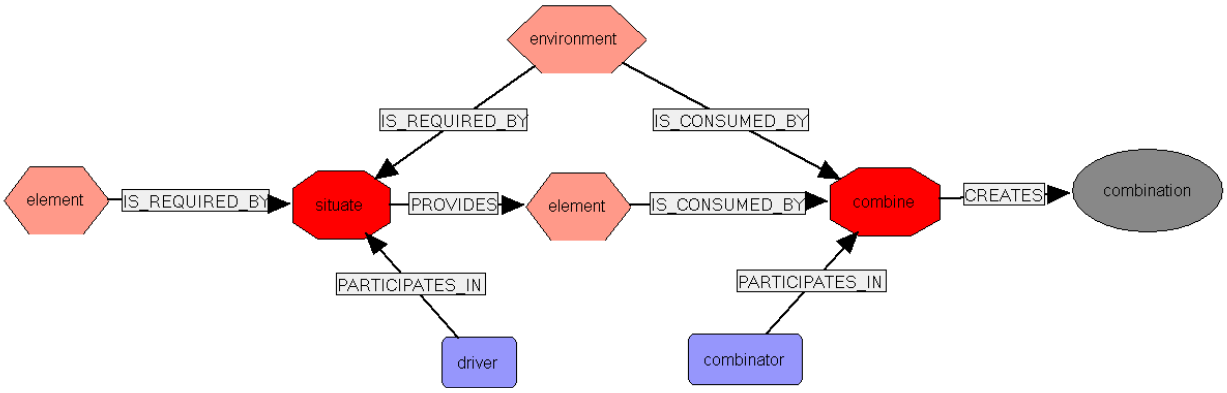


Figure 30: PSM *situate & combine*

Balance processes

In Nature, items put together tend to equilibrate each other's properties. The result of this kind of processes is an adjustment in the properties of the interacting elements until they reach an equilibrium point where changes stop

name	balance
goal	member(balanced set, e1) and member(balanced set, e2) and ¬ interact(e1, e2)
input roles	unbalanced set, stabilizer
output roles	balanced set

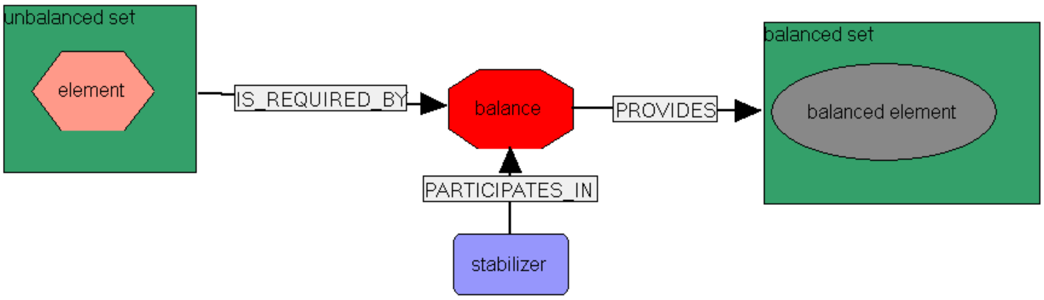


Figure 31: PSM *balance*

Examples of balance processes achievable by this PSM are osmosis and the autoionization of water.

Category *Locate*

Situate processes

This process category and its associated PSM refer to spatial notions. Though expressed in the Physics domain, using the MATH knowledge type, other domains, fundamentally Biology, express movement basically in terms of the PCS knowledge type, describing the cause and the agents inducing motions and their effects on the matter.

name	Situate
goal	original situation ≠ final situation and property(element, ep) and property(situated element, sp) and ep = sp
input roles	element, driver, original situation
output roles	situated element, final situation

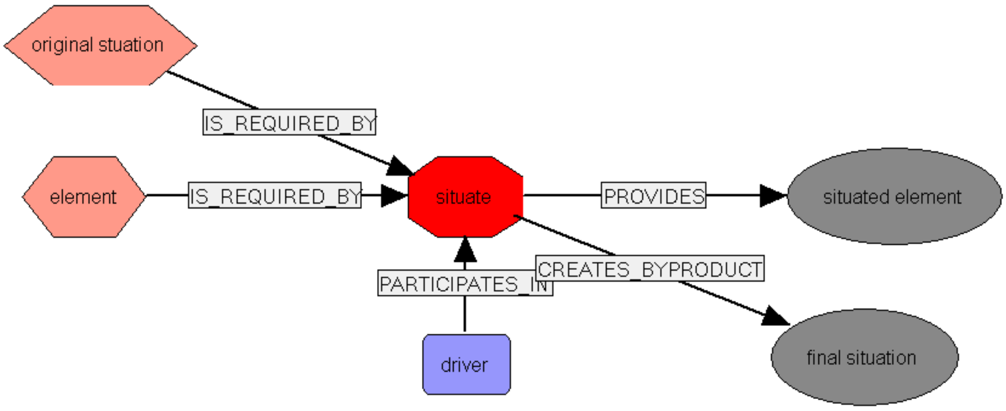


Figure 32: PSM *situate*

Examples of processes achievable by PSM *situate* are cell export/import, where substances are transported in and out of the cell, and cell movement, where it is the whole cell that moves from one location to another.

Oscillate processes

PSM *oscillate* specializes *situate*. Figure 33 adds a visual notion of control flow to the extended data flow view. Oscillation is periodic and needs specifying a loop condition in that regard by means of the specific construct of the process metamodel *periodic action*.

name	oscillate
goal	original situation = periodic situation property(element, ep) and property(situated element, sp) and ep = sp
input roles	element, driver, original situation
output roles	situated element, periodic situation

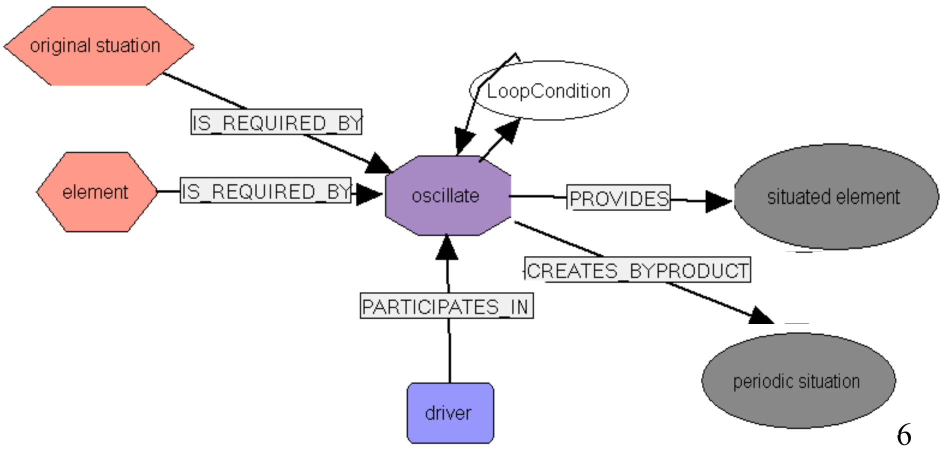


Figure 33: PSM *oscillate*

Wave propagation, particle oscillation, vibration and applications like echolocation in the Physics domain can be classified into this category and represented using PSM *oscillate*.

Rearrange processes

In general, it appeals to processes where elements are recombined but their internal structure remains. It can also be regarded as a specialization of transformation processes.

name	rearrange
goal	layout(rearrangement set, initial_layout) and layout (rearranged set, final_layout) and initial_layout ≠ final_layout
input roles	rearrangement set, driver
output roles	rearranged set

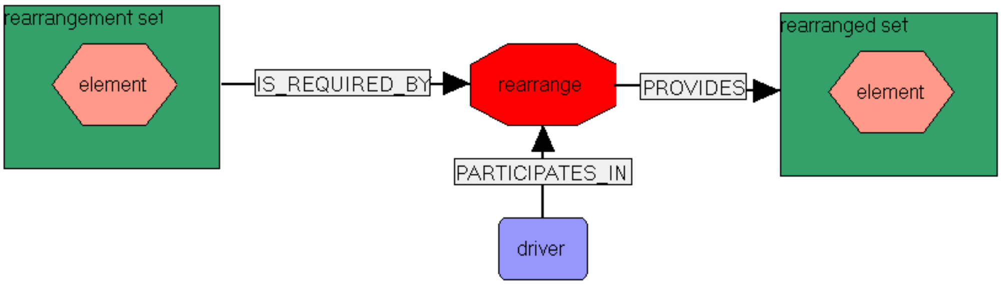


Figure 34: PSM *rearrange*

Examples of rearrangement include changes in conformation, i.e., processes leading not to a change of the chemical structure, but to a change in the (3-dimensional) conformation of a substance, e.g., a change in protein conformation, chromosome condensation, and contraction.

Release processes

In general, an agent acting upon a store for a set of items gives them out upon fulfilment of a precondition. PSM *accumulate & consume* can be applied to this process category by dividing the original process in two subprocesses: *accumulate* and *consume*. During the first subprocesses, elements of a given type are accumulated. Then, upon a certain condition, the accumulated elements are consumed. Examples of this process category include hormone secretion after accumulation in the corresponding gonads.

name	accumulate & consume
goal	member(storage, element) and is_byproduct_of (waste, element)
subactions	accumulate, consume
input action	accumulate
output action	consume
input roles	element, storage, accumulator, consumer
output roles	waste

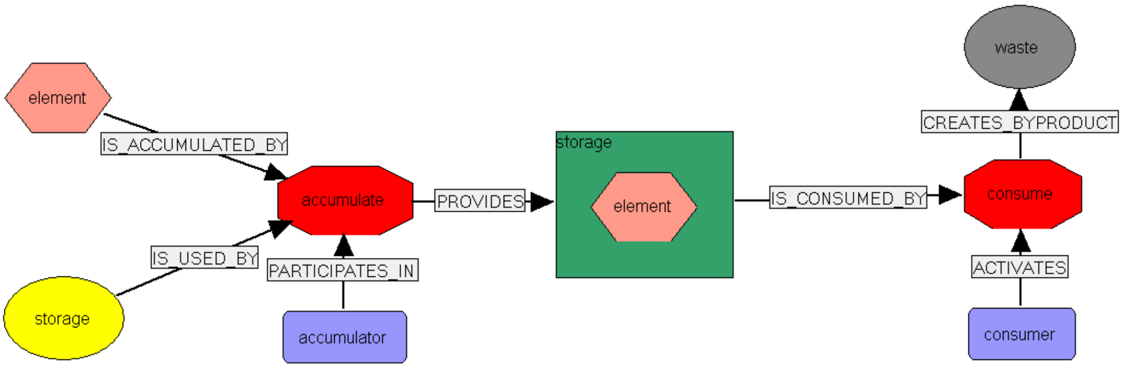


Figure 35: PSM *accumulate & consume*