9

A Knowledge Representation Formalism for Semantic Business Process Management

Ermelinda Oro and Massimo Ruffolo
High Performance Computing and Networking Institute of the National Research Council, Altilia srl
Italy

1. Introduction

Business process models are increasingly used to create clarity about the logical sequence of activities in public and private organizations belonging to different industries and areas. To improve *Business Process Management* (BPM), semantic technologies (like ontologies, reasoners, and semantic Web services) should be integrated in BPM tools in order to enable semantic BPM. Semantic Business Process Management (SBPM) approaches and tools aim at allowing more efficient and effective business process management across complex organizations. By semantic BPM decision makers can get transparent, fast, and comprehensive view of relevant business processes for better analyzing and driving processes. In defining semantic BPM tools aimed at improving the quality of process models and subsequent process analyses, a key aspect to take into account is to represent in combined way static knowledge regarding a specific application domain (i.e. domain ontologies) and dynamic knowledge related to process schemas and instances that are typically performed in a given domain. For example, in the health care domain, where the evidence-based medicine has contributed to define and apply clinical processes for caring a wide variety of diseases, a process-oriented vision of clinical practices may allow for enhancing patient safety by enabling better risks management capabilities.

In this Chapter is firstly summarized the large body of work currently available in the field of knowledge representation formalisms and approaches for representing and managing business processes. Then a novel ontology-based approach to business process representation and management, named Static/Dynamic Knowledge Representation Framework (SD-KRF), is presented. The SD-KRF allows for expressing in a combined way domain ontologies, business processes and related business rules. It supports semantic business process management and contributes to enhancing existing BPM solutions in order to achieve more flexible, dynamic and manageable business processes. More in detail, the presented framework allows methods for:

1. Creating ontologies of business processes that can be queried and explored in a semantic fashion.
2. Expressing business rules (by means of *reasoning tasks*) that can be used for monitoring processes.
3. Extracting information from business documents. Semantic information extraction allows the acquisition of information and metadata useful for the correct execution of business

processes from unstructured sources and the storage of extracted information into structured machine-readable form. Such a facility makes available large amount of data on which data mining techniques, can be performed to discover patterns related to adverse events, errors and cost dynamics, hidden in the structure of the business processes, that are cause of risks and of poor performances.

4. Querying directly enterprise database in order to check activity status.
5. Executing business processes and acquiring process instances by means of either *workflow enactment* (predefined process schemas are automatically executed) or *workflow composition* (activity to execute are chosen step-by-step by humans).
6. Monitoring business processes during the execution by running reasoning tasks.
7. Analyzing acquired business process instances, by means of querying and inference capabilities, in order to recognize errors and risks for the process and the whole organization.

SD-KRF is an homogeneous framework where the domain knowledge, the process structures, and the behavioral semantics of processes are combined in order to allow querying, advanced analysis and management of business processes in a more flexible and dynamic way.

2. Semantic business process management at a glance

BPM links processes and information systems. One of the most important aspect of BPM is the modeling of processes. Historically, process modeling has mainly been performed with general purpose languages, such as Activity Diagrams (AD), Business process Modeling Notation (BPMN) or Event-driven Process Chains (EPC). Such instruments are not suitable for an automated semantic process analysis because semantic modeling of structural elements and domain knowledge are missing. In recent years different languages and approaches for semantic business process management have emerged. In this Section will be briefly described languages for representing processes and their semantics.

By considering the abilities of representing business processes, as described in van der Aalst (2009), existing languages for processes modeling can be classified in:

- **Formal languages**. Processes are described by using formal models, for examples Markov chains and Petri nets. Such languages have unambiguous semantics.
- **Conceptual languages**. Processes are represented by user-friendly semi-formal languages. Example of well known conceptual languages are UML activity diagrams, BPMN (Business Process Modeling Notation), and EPCs (Event- Driven Process Chains). Activity diagrams (or control flow diagrams) is a type of UML (unified modeling language OMG (2011)) diagrams. They provide a graphical notation to define the sequential, conditional, and parallel composition of lower-level behaviors, therefore they are suitable for modeling business processes. The Event-driven Process Chain (EPC) van der Aalst (1999) is a type of flowchart used for business process modeling and compared to UML activity diagrams, the EPC covers more aspects such as a detailed description of business organization units together with their respective functions as well as information and material resources used in each function. These essential relationships are not explicitly shown in activity diagrams. The Business Process Model and Notation (BPMN) White (2006) is a graphical notation for drawing business processes, proposed as a standard notation. The language is similar to other informal notations such as UML activity diagrams and extended

event-driven process chains. Models expressed in terms of BPMN are called Business Process Diagrams (BPDs). A BPD is a flowchart having different elements: Flow Objects, Connecting Objects, Swim-lanes, Artifacts, Events, Activities, and Gateways. Events are comparable to places in a Petri net, in fact they are used to trigger and/or connect activities. Whereas in UML Activity Diagrams and in BPMN resource types are captured as swim-lanes, with each task belonging to one or more swim-lane, in Event-driven Process Chains (EPC) resource types are explicitly attached to each task. These type of languages describe only the desired behavior of processes, and do not have a formal semantics, therefore they are not suitable for enabling processes execution.

- **Execution languages**. Because formal languages are too general and conceptual languages are aiming at the representation of processes and not directly at execution, languages that consider the processes enactment have been defined. The most common language in this category is BPEL (Business Process Execution Language) Wohed et al. (2006). BPMN diagrams are refined into BPEL specifications, but such translation is a difficult task because BPMN lacks of formal semantics. Therefore, several attempts have been made to provide semantics for a subset of BPMN Weske (2007). Other proprietary enactment languages have been defined. For example, XPDL XPDL (2011) is a very common language based on BPMN.

2.1 Semantic business process management

BPM is a difficult task because the semantic of a business processes is frequently hidden in complex models obtained by different description and enactment languages. The explicit representation of domain knowledge related to business processes combined to explicit description of the semantic processes could help to obtain advices, alerts, and reminders. Furthermore, reasoning capabilities allow for representing and managing business rules and better enacting and monitoring of processes Peleg (2009). Classical languages adopted for representing process models provide a low degree of automation in the BPM lifecycle. In particular, there are many difficulties in the translations of business modeling (performed by business expert analist) to workflow models (which are executable IT representations of business processes). Like Semantic Web Services achieve more automation in discovery and mediation with respect to conventional Web services, BPM systems can obtain more automation by using knowledge representation and reasoning, and therefore semantic technologies Hepp et al. (2005).

Initially, knowledge representation and reasoning is been used for artificial intelligence tasks Newell (1980) to support humans in decision making. Then, rule-based systems were introduced. An important example is Mycin Shortliffe (1976) that represented clinical knowledge and contained if-then-else rules in order to derive diagnoses and treatments for a given disease. After, it was integrated database for representing knowledge in decision support systems. An important example is the Arden System for Medical Logic Modules (MLMs) Hripcsak et al. (1994) system. MLMs, in Arden Syntax, define decision logic via a knowledge category that has data, event, logic, and action slots useful in representing processes. Finally, ontologies Gruber (1995) were used for formally representing the knowledge as a set of concepts within a domain, and the relationships between those concepts. An ontology may be used to describe the domain, and to reason about the entities and relations within that domain in order to provide decision support. It is noteworthy that to add, delete or modify knowledge in rule-based systems was a difficult task, whereas ontologies

are more simply modifiable. Ontologies and Semantic Web service technologies can be used throughout the BPM lifecycle Hepp et al. (2005); Wetzstein et al. (2007).

The use of semantics in BPM creates Semantic Business Process Management (SBPM) System. The goal of Semantic Business Process Management is to achieve more automation in BPM by using semantic technologies. In Wetzstein et al. (2007) the SBPM lifecycle is described. There are 4 principal phases: *Process Modeling*, *Process Implementation*, *Process Execution*, and *Process Analysis*. The usage of semantic technologies increases the automation degree and the BPMS functionalities.

During the *process modeling phase*, the annotation of business process models allows for associating semantics to task and decisions in the process. The annotation is usually performed by using ontologies that describe domains or processes components. Generally, ontologies are created by ontology engineers, domain experts and business analysts. Different types of ontologies are relevant to business process management Hepp & Roman (2007). For instance, an organizational ontology is used to specify which organizational tasks have to be performed, in combination with a Semantic Web Service (SWS) ontology that specify the IT services that implement tasks, and domain ontologies that describe data used in the processes. The processes annotation enables additional semantics functionalities. In fact, ontological annotation of tasks enables the reuse of process fragments in different business processes in the *implementation phase*. During the *execution phase*, semantic instances are created, semantic checks of obtained instances can be automatically evaluated by calling reasoning tasks. During the semantic BP *analysis phase*, two different features are distinguished: (i) process monitoring which aims at providing relevant information about running process instances in the process execution phase, (ii) process mining that analyzes already executed process instances, in order to detect points of improvement for the process model. Such features take advantages by the semantic annotation. For instance, business analysts can formulate semantic queries and use reasoning to deduce implicit knowledge. Analysis allows for improving business processes for decreasing costs or risks in processes executions Medeiros & Aalst (2009).

There exist a lot of work addressing the enhancement of Business Process Management Systems ter Hofstede et al. (2010) by using Semantic Web techniques and, in particular, computational ontologies Hepp et al. (2005).

Robust and effective results have been obtained from European research projects CORDIS (2011), such as SUPER SUPER (2011), PLUG-IT PLUG-IT (2011) and COIN COIN (2011). They implemented new semantics-based software tools that enhance BPs of companies, and lower costs and risks. SUPER SUPER (2011), that means Semantics Utilised for Process Management within and between Enterprises, is an European Project financed from the Europen Union 6th Framework Program, within Information Society Technologies (IST) priority. The project successfully concluded at 31st of March 2009. The objective of SUPER was to make BPM accessible for business experts adding semantics to BP. Semantic Web and, in particular, Semantic Web Services (SWS) technology allows for integrating applications at the semantic level. Based on ontologies, Semantic Web (SW) technologies provide scalable methods and tools for the machine readable representation of knowledge. Semantic Web Services (SWS) use SW technologies to support the automated discovery, substitution, composition, and execution of software components (Web Services). BPM is a natural application for SW and SWS technology. The project SUPER combines SWS and BPM, provided a semantic-based and context-aware framework, and created horizontal ontologies which describe business

processes and vertical telecommunications ontologies to support domain-specific annotation. SUPER ontologies allow telecoms business managers to search existing processes, to model new business processes, to modify process models, to search for semantic web services that compose business processes and to execute implemented business process models.

The project plugIT PLUG-IT (2011), that means Plug Your Business into IT, has been co-funded by the European Union Intelligent Content and Semantics. It is based on the observation of the necessity to align Business and Information Technology (IT). The result of the project was the IT-Socket Knowledge portal. The project is based on the idea that IT can be consumed by plugging in business, like electric power is consumed when plugging in electronic devices in a power socket. ITSocket externalizes the expert knowledge by using graphical semi-formal models, such a knowledge is formalized in order to enable a computer-supported alignment using semantic technologies. Figure 1 shows business and IT experts that formalize the knowledge and so they enable automated support of business and IT alignment. In particular the alignment can be delegated to semantic technologies.

COIN, that means Enterprise COllaboration and INteroperability, COIN (2011) is an integrated project in the European Commission Seventh Framework Programme, that stars in 2008 and ends in 2011. The scope of the project is create a pervasive and self-adapting knowledge that enable enterprise collaboration and interoperability services in order to manage and effectively operate different forms of business collaborations.

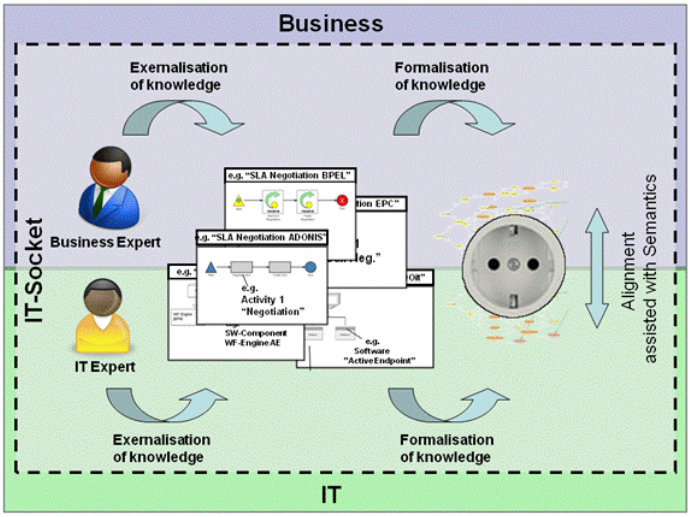


Fig. 1. IT-Socket for business and IT alignment

In literature a lot of ontology-based approaches to business process management have been proposed. Initially, a set of approaches was proposed to apply techniques borrowed from the Semantic Web to the BP management context SUPER (2011). In Missikoff et al. (2011) an ontology-based approach for querying business process repositories for the retrieval of process fragments to be reused in the composition of new BPs is presented. The proposed solution is composed by an ontological framework (OPAL) aimed at capturing the semantics of a business scenario, and a business process modelling framework (BPAL) to represent the workflow logic of BPs. In Markovic (2008) a querying framework based on ontologies is

presented. In Francescomarino & Tonella (2008) a visual query language for business Process is described. Processes are represented trough a BPMN meta-model ontology annotated by using domain ontologies, SPARQL queries are visually formulated.

Other approaches based on meta-model ontologies have been presented in Haller et al. (2008; 2006) In Hornung et al. (2007) the authors present an initial idea for an automatic approach for completion of BP models. Their system recommends appropriate completions to initial process fragments based on business rules and structural constraints. The main elements are modeled by using an OWL representation of Petri nets that allows for efficiently computing the semantic similarity between process model variants. Additionally the approach makes use of the Semantic Web Rule Language (SWRL), which is based upon a combination of OWL DL with Unary/Binary Datalog RuleML Boley et al. (2001), to model additional constraints imposed by business rules. Ontoprocess system Stojanovic & Happel (2006) for semantic business process management semantically described, business processes are combined with SWRL rules by using a set of shared ontologies that capture knowledge about a business domain. These formal specifications enable to automatically verify if a process description satisfies the consistency constraints defined by business rules. In order to query BPs, some graph-matching-based approaches have been proposed Awad et al. (2008); Haller et al. (2006) . In Awad et al. (2008) BPs are compiled to finite state models, so model checking techniques allow for verifying structural features of process schemas. However, the semantics of the business domain is not considered. Other approaches that allow for modeling and reasoning over workflows are based on logic programming Montali et al. (2008); Roman & Kifer (2007) have been introduced. Such approaches allow for checking and enacting BPs, but they are not used for querying.

As shown, a lot of approaches have been proposed in litterature, but no one is capable to semantically manage in a comprehensive way all phases of SBPM lifecycle.

2.2 Business process management in the health care domain

The health care domain is of great interest for BPM. In fact, in the recent past, a strong research effort has been taken to provide standard representations of both declarative and procedural medical knowledge. In particular, in the area of medical knowledge and clinical processes representation, there exist one of the most rich collection of domain ontologies available worldwide and a wide variety of formalisms for clinical process representation. In the following, available approaches and systems for medical ontologies and clinical process representation and management are described.

A very famous and widely adopted medical thesaurus is Mesh, the Medical Subject Headings classification MESH (2011). It provides a controlled vocabulary in the fields of medicine, nursing, dentistry, veterinary medicine, etc. MeSH is used to index, catalogue and retrieve the world's medical literature contained in PubMed. Another classification, that has become the international standard diagnostic classification for all medical activities and health management purposes, is ICD10-CM ICD (2011); WHO (2011) the International Classification of Diseases Clinical Modification, arrived to its 10th Revision. The most comprehensive medical terminology developed to date is SNOMED-CT SNOMED (2011), the Systematized Nomenclature of Medicine Clinical Terms, based on a semantic network containing a controlled vocabulary. Electronic transmission and storing of medical knowledge is facilitated by LOINC, the Logical Observation Identifiers Names and Codes LOINC (2011), that consists in a set of codes and names describing terms related to clinical laboratory results, test results

and other clinical observations. Machine-readable nomenclature for medical procedures and services performed by physicians are descried in CPT, the Current Procedural Terminology CPT (2011), a registered trademark of the American Medical Association. A comprehensive meta-thesaurus of biomedical terminology is the NCI-EVS NCI-EVS (2011) cancer ontology. Some medical ontologies are, also, due to European medical organizations. For example, CCAM the Classification Commune des Actes Medicaux CCAM (2011), is a French coding system of clinical procedures that consists in a multi-hierarchical classification of medical terms related to physician and dental surgeon procedures. A classification of the terminology related to surgical operations and procedures that may be carried out on a patient is OPCS4, the Office of Population Censuses and Surveys Classification of Surgical Operations and Procedures 4th Revision OPCS-4 (2011), developed in UK by NHS. The most famous and used ontology in the field of healthcare information systems is UMLS, the Unified Medical Language System UMLS (2011), that consists in a meta-thesaurus and a semantic network with lexical applications. UMLS includes a large number of national and international vocabularies and classifications (like SNOMED, ICD-10-CM, and MeSH) and provides a mapping structure between them. This amount of ontologies constitutes machine-processable medical knowledge that can be used for creating semantically-aware health care information systems.

The evidence-based medicine movement, that aims at providing standardized clinical guidelines for treating diseases Sackett et al. (1996), has stimulated the definition of a wide set of approaches and languages for representing clinical processes. A well known formalisms is GLIF, the Guideline Interchange Format GLIF (2011). It is a specification consisting of an object-oriented model that allows for representing sharable computer-interpretable and executable guidelines. In GLIF3 specification is possible to refer to patient data items defined by a standard medical vocabularies (such as UMLS), but no inference mechanisms are provided. Pro*forma* Sutton & Fox (2003) is essentially a first-order logic formalism extended to support decision making and plan execution. Arden Syntax HL7 (2011); Peleg et al. (2001); Pryor & Hripcsak (1993) allows for encoding procedural medical knowledge in a knowledge base that contains so called Medical Logic Modules (MLMs). An MLM is a hybrid between a production rule (i.e. an "if-then" rule) and a procedural formalism. It is less declarative than GLIF and Pro*forma*, its intrinsic procedural nature hinders knowledge sharing. EON Musen et al. (1996) is a formalism in which a guideline model is represented as a set of scenarios, action steps, decisions, branches, synchronization nodes connected by a "followed-by" relation. EON allows for associating conditional goals (e.g. if patient is diabetic, the target blood pressures are 135/80) with guidelines and subguidelines. Encoding of EON guidelines is done by Protégé-2000 Protégé (2011) knowledge-engineering environment.

3. Static/dynamic knowledge representation framework

The key idea which the Static and Dynamic Knowledge Representation Framework (SD-KRF) is based on is that elements of the workflow meta-model (i.e. processes, nodes, tasks, events, transitions, actions, decisions) are expressed as ontology classes Oro & Ruffolo (2009); Oro et al. (2009b). This way workflow elements and domain knowledge can be easily combined in order to organize processes and their elements as an ontology. More in detail, the SD-KRF allows for representing extensional and intensional aspects of both declarative and procedural knowledge by means of:

- *Ontology and Process Schemas*. The former expresses concepts related to specific domains. Ontology contents can be obtained by importing other existing ontologies and thesaurus

or by means of direct manual definition. The latter are expressed according with the workflow meta-model illustrated in Section 3.1.

- *Ontology and Process Instances* both expressed in term of ontology instances. In particular, ontology class instances can be obtained by importing them from already existing ontologies or by creating them during process execution. Process instances are created exclusively during process execution. Instances are stored in a knowledge base.
- *Reasoning Tasks* that express, for instance, decisions, risks and business rules.
- *Concept Descriptors* that express information extraction rules capable to recognize and extract ontology instances contained in unstructured documents written in natural language. By concept descriptors ontology instances can be automatically recognized in documents and used for both enriching the knowledge base and annotating unstructured documents related to given domains and processes.

The SD-KRF constitutes an innovative approach for semantic business process management in the field of healthcare information systems. Main features of the presented semantic approach (founded on logic programming) is that it, conversely to already existing systems and approaches, enables to represent process ontologies that can be equipped with expressive business rules. In particular, the proposed framework allows for jointly managing declarative and procedural aspects of domain knowledge and express reasoning tasks that exploit represented knowledge in order to prevent errors and risks that can take place during processes execution. The framework enables, also: (i) manual process execution in which each activity to execute in a given moment is chosen by a human actor on the base of the current configuration of patient and disease parameters, and (ii) automatic execution by means of the enactment of an already designed process schema (e.g. guidelines execution). During process execution, process and ontology instances are acquired and stored in a knowledge base. The system is able to automatically acquire information from electronic unstructured medical documents, exploiting a semantic information extraction approach. Extracted information are stored into a knowledge base as concept instances. The processes execution can be monitored by running (over clinical process schemas and instances) reasoning tasks that implements business rules.

3.1 Modelling process

A significant amount of research has been already done in the specification of mechanisms for process modeling (see, Georgakopoulos et al. (1995) for an overview of different proposals). The most widely adopted formalism is the control flow graph, in which a workflow is represented by a labeled directed graph whose nodes correspond to the activities to be performed, and whose arcs describe the precedences among them. In the SD-KRF we adopt the graph-oriented workflow meta-model shown in Figure 2.a and 2.b, inspired by the JPDL JPDL (2011) process modeling approach. The adopted meta-model: (i) covers the most important and typical constructs required in workflow specification; (ii) allows for executing processes in the SD-KRF by using the JBPM workflow engine; (iii) allows for using workflow mining techniques grounded on graph-oriented meta-models.

Since our scope is to allow the semantic representation of processes, we need firstly to formally define the process meta-model as the following 6-tuple:

$$\mathcal{P} = \langle N, A_r, E_v, A_n, T_k, E \rangle$$

where:

- N is a finite set of *nodes* partitioned in the following subsets: task nodes N_T (that represent activities in which a humans or machines perform tasks), subprocess nodes N_{SP} (that model activities referring processes external to the current one), group nodes N_G (that represent a set of nodes that can be executed without a specific order), custom nodes N_C (that model activities in which custom methods can be executed and handled automatically), wait nodes N_W (that represent activities that temporary stop the execution while they execute methods), join nodes N_J, and fork nodes N_F (that are respectively used to combine or split execution paths) and decision nodes N_D (that allow for controlling the execution flow on the base of conditions, variables or choices performed automatically or by human actors).
- A_r is a set of *actors*. Actors can be human or automatic. They represent the agents that execute a given task or activity.
- A_n is a set of *actions*. An action is a special activity that can be performed as answer to the occurrence of an event.
- T_k is a set of *tasks* that represent tasks to execute in task nodes.
- $E = \{\langle x, y\rangle : x \in N_{From} \wedge y \in N_{To}\}$ is a set of *transitions* in which the following restrictions hold, when $N_{From} \equiv N_{FN} \cup N_D$ then $N_{To} \equiv N_{FN} \cup N_{FCN} \cup N_{end}$ and when $N_{From} \equiv N_{FCN}$ then $N_{To} \equiv N_{FN} \cup N_D$. Moreover, for each process there is a transition of the form $e_{start} = \langle N_{start}, y\rangle$ where $y \in N_{FN}$ and one of the form $e_{end} = \langle x, N_{end}\rangle$ where $x \in \{N_{FN} \cup N_D\}$. The subset $E_d \subset E$ where $E_d = \{\langle x, y\rangle : x \in N_D \wedge y \in N_{FN}\}$ is the set of *decisions*. A decision relates a decision node to a flow node and could hold a *decision rule* that is used at run-time to automatically control the execution flow of a process.
- E_v is a set of *events*. An event causes the execution of an action that constitutes the answer to the event. An event can be, for example, the throwing of an exception during the execution of a task.

3.2 Modeling static and dynamic knowledge

Formally the Static/Dynamic Knowledge Representation Framework (SD-KRF) is the 5-tuple having the following form:

$$\mathcal{O} = \langle D, A, C, R, I\rangle.$$

Where ontology/process schemas are expressed by using elements of D, A, C and R in $\mathcal{O}$ that are *finite* and *disjoint* sets of entity names respectively called *data-types*, *attribute-names*, *classes* and *relations*. The set of classes C is organized in taxonomies and partitioned in two subsets:

- The set of process classes $C_P = N \cup A_r \cup A_n \cup T_k \cup E_v$ that represents elements of the workflow meta-model. It is constituted by the union of classes representing nodes, actors, actions, tasks and events.
- The set of ontology classes C_O that represent concepts related to a specific knowledge domains.

The set R is a set of ontological relations partitioned in two subsets: the set of transition $R_P = E$, and the set of relations R_M used for representing relations between ontology concepts. In the following meaning and usage of $\mathcal{O}$ is explained by describing the implementation of a running example.

4. An example: representing ontologies and a process schemas in the medical domain

The medical domain offers many ontologies and thesaura describing diseases, drugs, medical examinations, medical treatments, laboratory terms, anatomy, patients administration, and clinical risks. Examples of medical ontologies are UMLS UMLS (2011), LOINC LOINC (2011), ICD10-CM ICD (2011), SNOMED SNOMED (2011). Many of such ontologies can be freely obtained from international organization that maintain them and can be automatically imported in the SD-KRF or manually entered by means of direct manual definition.

This section describes a clinical process for caring the breast neoplasm (Figure 2.c). Such an example in the medical domain, that will be used in the rest of the chapter as running example, allows for describing the ability of the SD-KRF to enable the representation of ontologies representing in combined way process schemas, ontological concepts and relations, and instances of both processes and ontologies.

The example considers practices carried out in the oncological ward of an Italian hospital, hence it is not general but specific for the domain of the considered ward. The clinical process is organized in the following 10 activities:

1. Task node *Acceptance* models patient enrollment. A patient arrives to the ward with an already existing clinical diagnosis of a breast neoplasm. This activity can be performed manually by an oncologist that collects patient personal data, and directly acquiring information from electronic medical records in natural language. The information extraction task is performed by exploiting the semantic information extraction approach described in Section 5. Extracted information are stored as ontology instances Oro et al. (2009a).
2. Group node *Anamnesis* represents a set of anamnesis activities: *general anamnesis* in which physiological general data (e.g. allergies, intolerances) are being collected; *remote pathological anamnesis*, concerning past pathologies; *recent pathological anamnesis*, in which each data or result derived from examinations concerning the current pathologies are acquired. These activities can be manually executed without a specific order or by exploiting semantic extraction rules (descriptors) that enable the recognition of information into unstructured source like pre-existing EMR.
3. Task node *Initial clinical evaluation* allows for acquiring the result of an examination of the patient by an oncologist.
4. Decision node *More clinical test requested* represents the decision to perform or not additional examination on the patient.
5. Group node *Other exams* models possible additional clinical tests. If requested these tests are conducted to find out general or particular conditions of patient and disease not fully deducible from the test results already available.
6. Task node *Therapeutic strategy definition* models the selection of a guideline with related drug prescription. At execution time the physician picks a guideline (selected among the guidelines already available in the knowledge base) that depends upon actual pathology state as well as other collected patient data.
7. Task node *Informed agreement sign* models the agreement of the patient concerning understanding and acceptance of consequences (either side effects or benefits) which may derive from the chosen chemotherapy, and privacy agreements.

8. Sub-process *Therapy administration*, models a subprocess that constitutes the guideline to execute for caring the patient.
9. Decision node *Therapy ended* models a decision activity about effects of the therapy and the possibility to stop or continue cares.
10. Task node *Discharging* models the discharging of the patient from the ward end allows for acquiring final clinical parameter values.

In the activities (6) and (8) risk and error conditions can be identified. At each guideline, chosen in (6), corresponds a prescription of drugs (chemotherapy). Hence the computation of doses, which may depend on patient's biomedical parameters such as body's weight or skin's surface, is required. Cross-checking doses is fundamental here, because if a wrong dose is given to the patient the outcome could be lethal. Furthermore, therapy administration ((8)-th activity) must contain checks that aims at verify type and quantity of chemotherapeutic drugs to submit to the cared patient.

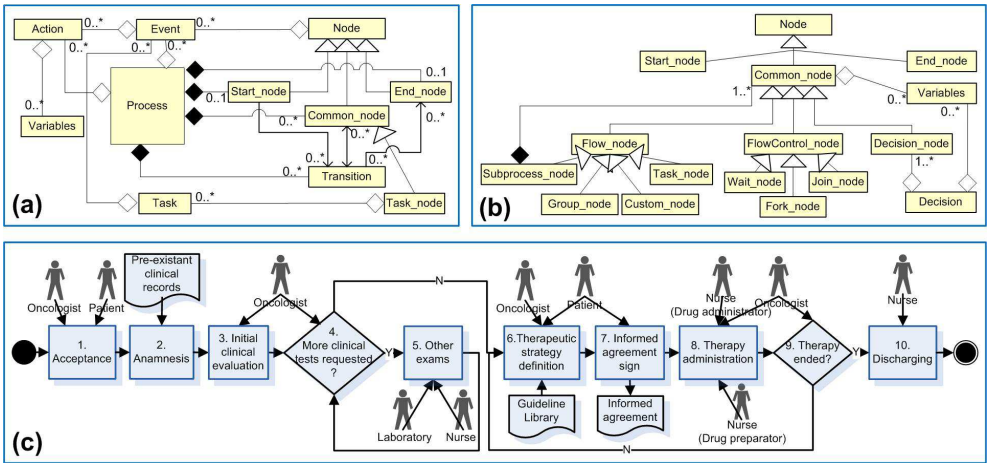


Fig. 2. (a) The process meta-model. (b) The nodes hierarchy. (c) A clinical process for caring the breast neoplasm

4.1 Ontology and process schemas

This section presents the syntax of the SD-KRF language by example. A class in $C \in \mathcal{O}$ can be thought as an aggregation of individuals (objects) that have the same set of properties (attributes $A \in \mathcal{O}$). From a syntactical point of view, a class is a name and an ordered list of attributes identifying the properties of its instances. Each attribute is identified by a name and has a type specified as a data-type or class.

In the following the implementation of the workflow meta-model in the SD-KRF language is firstly presented. In particular, *nodes* in C_P are implemented by using the class hierarchy (built up by using `isa` key-word) shown below.

```
class process(name:string).
class node(name:string, container: process, start_time:integer,
           end_time:integer).
```

```
    class start_node() isa{node}.
    class end_node() isa{node}.
    class common_node () isa{node}.
        class flowControl_node() isa{common_node}.
            class fork() isa{flowControl_node}.
            class join() isa{flowControl_node}.
            class wait_node() isa{flowControl_node}.
        class flow_node() isa{common_node}.
            class task_node(tasks:[task], handler:human_actor)
                        isa{flow_node}.
            class custom_node(handler: automatic_actor, method:string)
                        isa{flow_node}.
            class group_node(nodes:[node]) isa{flow_node}.
            class sub_process_node(sub_proc: process) isa{flow_node}.
        class decision_node(handler:actor) isa{common_node}.
            class automatic_decision_node(handler:automatic_actor)
                        isa{decision_node}.
            class manual_decision_node(task:task, handler:human_actor)
                        isa{decision_node}.
```

Task nodes and manual decision nodes contain *tasks* that are performed by humans. Tasks `class task(name: string).` collects values of activity variables given in input by human actor. *Actors* of a process (that can be human or automatic) represent the agents that execute a given task. They are represented by means of the following classes in C_P:

```
class actor(name:string).
    class human_actor() isa {actor}.
    class automatic_actor(uri:string) isa {actor}.
```

During the process enactment, by running risk and business rules, *events* may occur. Furthermore, an event can be generated by an exception during the execution of a task. Events, and related actions to performs in response, are represented in C_P by the following classes.

```
class event(relativeTo:object, timestamp:integer).
    class node_event(relativeTo:node) isa{event}.
    class task_event(relativeTo:task) isa{event}.
    class process_event(relativeTo:process) isa{event}.
class action(method:string).
```

Relationships among objects are represented by means of relations, which like classes, are defined by a name and a list of attributes. *Transitions* and *decisions*, in R_P, that relate couple of nodes, are represented by means of the following ontology relations.

```
relation transition(name:string, from:node, to:node).
relation decision(name:string, from:decision_node, to:node).
```

When the user defines a specific process schema s/he can specialize original meta-model elements for adding new semantic attribute required by the specific process. In the following are shown some classes representing nodes of the running example depicted in Figure 2.

```
class acceptance_node(tasks:[acceptance_form],handler:physician)
      isa{task_node}.
class anamnesis_node(nodes:[general_anamnesis_node,
      remotePathological_anamnesis_node, recentPathological_anamnesis_node])
      isa {group_node}.
class recentPathological_anamnesis_node(tasks:[pathology_form],
      handler:physician) isa {task_node}.
class therapeutic_strategy_definition_node(tasks:[therapeutic_strategy_form],
      handler:nurse) isa {task_node}.
class therapy_administration_node(sub_process:therapy_administration_process)
      isa{sub_process_node}.
class more_tests_node(task:more_tests_form) isa{manual_decision_node}.
```

`acceptance` and `therapeutic_trategy_definition` process activities are represented as subclasses of `task_node` class, in fact they represent activities in which tasks consist in the execution of forms filled by humans. Whereas `anamnesis_node`, which `Recent Pathological anamnesis` activity belongs to, is represented as a subclass of `group_node` class. `therapy_administration_node` and `more_tests_node` are specializations of `sub_pr oc_node` and `decision_node` respectively. Human actors that operate in this clinical process could be physicians, nurses and patients. They are represented by a person hierarchy that exploits multiple inheritance capabilities in order to express that persons are also human actors of the clinical process.

```
class person(fiscalCode:string,name:string,surname:string,sex:sex_type,
            bornDate:date,address:address).
    class patient(hospitalCard:string, weight:float, heigthCm:float)
            isa {person,human_actor}.
    class healthCareEmploy(occupation:string, role:string)
            isa {person,human_actor}.
        class nurse() isa {healthCareEmploy}.
        class physician() isa {healthCareEmploy}.
```

Class schemas representing tasks related to task-nodes can be expressed by using the following class schemas. Attribute types can be classes represented in C_M expressing different medical concepts (e.g. diseases, drugs, body parts). During task execution values of resulting class instances are obtained from fields filled in forms.

```
class task(name: string).
    class acceptance_form(patient:patient, acc_date:date) isa{task}.
    class pathology_form(disease:disease) isa{task}.
    class chemotherapeutic_strategy_form(strategy:therapeuticStrategy)
          isa{task}.
    class more_tests_form(choice:boolean)isa{task}.
```

In a clinical process, an event can be activated by an exception during the execution of a node or by a reasoning task aimed at control business rules. A reasoning task checks parameters values of running node and already acquired node instances and throws an event related to an error. An example of different kinds of possible errors is shown in the following taxonomy, where the attribute `msg` of the class `view_msg` (action) is the message to display when the error occurs.

```
class task_event(relativeTo:task) isa{event}.
    class medicalError(msg:string) isa{task_event}.
        class drugPrescriptionError() isa {medicalError}.
class view_msg(msg:string) isa {action}.
```

Class schemas in C_M expressing knowledge concerning anatomy, breast neoplasm disease and related therapies and drugs have been obtained (imported) from the Medical Subject Headings (Mesh) Tree Structures , the International Classification of Diseases (ICD10-CM) . and the Anatomical Therapeutic Chemical (ATC/DDD) .

```
class anatomy(name:string).
    class bodyRegion() isa {anatomy}.
class disease(descr:string).
  class neoplasm() isa {disease}.
    class malignant_neoplasm() isa {neoplasm}.
       class primarySited_neoplasm(site:bodyRegion,zone:string)
             isa {malignantNeoplasm}.
          class breast_primarySited_neoplasm() isa {primarySited_neoplasm}.
class drug(name:string, ddd:float, unit:unitOfMeasure,admRoute:[string],
          notes:string).
    class antineoplasticAndImmunomodulatingAgent() isa {drug}.
        class endocrineTherapy() isa {antineoplasticAndImmunomodulatingAgent}.
            class hormoneAntagonistsAndRelatedAgents()isa {endocrineTherapy}.
                class enzymeInhibitors()
                      isa {hormoneAntagonistsAndRelatedAgents}.
            class hormoneAndRelatedAgents()isa {endocrineTherapy}.
                class estrogens() isa {hormoneAndRelatedAgents}.
class code(c:string).
    class icd10Code(chapter:integer, block:string,category:string,
          subCat:string) isa {code}.
    class mesh08Code(category:string, subCat:string) isa {code}.
class therapy(name:string, dru:drug, dose:float).
class therapeuticStrategy(patient:patient, therapy:therapy,startDate:date,
                          nDay:integer).
```

The previous classes are a fragment of a medical ontology inherent (breast) neoplasm cares and are used to model the clinical process shown in Section 4. Class `primarySited_neoplasm` shows the ability to specify user-defined classes as attribute types (i.e. `site:bodyRegion`). Class `drug` has a list-type attribute `admRoute:[string]` representing possible route of administration for a drug (for example inhalation, nasal, oral, parenteral). Relation schemas expressing medical knowledge can be declared by using the following syntax:

```
relation suffers (patient:patient, disease:disease).
relation relatedDrug (dis:disease, dru:drug).
relation sideEffect (dru:drug, effect:string).
relation classifiedAs (dis:disease, c:code).
```

Relation `suffer` asserts diseases suffered by a patient. Relations `relatedDrug` and `sideEffect` associates respectively drugs to a diseases and side effects to drugs. Moreover, relation `classifiedAs` enables users to query the ontologies by using codes defined in the original medical ontologies.

4.2 Ontology and process instances

Clinical process instances are expressed by ontology instances and created exclusively during process execution. Classes instances (objects) are defined by their *oid* (that starts with #) and a list of attributes. Instances obtained by executing the running example, are shown in the following.

```
#1:neoplasm_process(name:"Breast Neoplasm").
#2:therapy_administration_process(name:"Therapy Administration").
#1_1:acceptance_node(name:"Acceptance", container:#1, start_time:6580,
    end_time:16580, tasks:[#1_1_1], handler:#27).
#1_2:anamnesis_node(name:"Anamnesis", container:#1, start_time:16570,
    end_time:26580, nodes:[#1_2_1, #1_2_2, #1_2_3])
#1_2_3:recentPathological_anamnesis_node(name:"Recent Pathological Anamnesis",
       container:#1, start_time:19580, end_time:26570,tasks:[#1_2_3_1],
       handler:#27).
...
```

As described in section 4, instance of `anamnesis_node #1_2` is composed by a set of anamnesis activities represented by means of their *id*. The object `#1_2_3` belongs to `#1_2`. Objects `#1_1`, `#1_2_3` are tasks executed in custom and manual decision node and are stored as their attributes. When execution arrives in a task node or in a manual decision node, task instances are created and the user input is stored as values of the task attributes. Some tasks related to task nodes are shown in the following.

```
#1_1_1:acceptance_form(name:"Acceptance", patient:#21, acc_date:#data_089).
#1_2_3_1:pathology_form(name:"Recent Pathology", disease:#neoB_01).
```

For example, the instance `1_1_1` of the class `acceptance_form` is created by an oncologist that fills in a form. It contains an instance of patient class.

Transition and decision tuples, created during the process execution, are shown in the following. The tuple of `decision` in the following example is obtained as a manual choice of an oncologist, but instances of decisions could be automatically generated by means of reasoning tasks.

```
transition(name:"Acceptance-Anamnesis",from:#1_0, to:#1_1).
decision(name:"More Clinical Tests requested - No",from:#1_4, to:#1_6).
```

Instances of the classes `bodyRegion`, `breast_primarySited_neoplasm`, and of the subclasses of `drug` and `code`, can be obtained by importing them from already existing medical ontologies and can be declared as follows:

```
#A01.236: bodyRegion(name:"breast").
#neoB_01: breast_primarySited_neoplasm(descr:"Malignant neoplasm of breast",
          site:#A01.236, zone:"Nipple and areola").
#L02BG03: enzymeInhibitors(name:"Anastrozole", ddd:1, unit:mg,
          admRoute:["oral"], notes:"").
#L02AA04: estrogens(name:"Fosfestrol", ddd:0.25, unit:g,
          admRoute:["oral","parenteral"], notes:"").
#icd10_C50.0: icd10Code(c:"C50.0", chapter:2, block:"C", category:"50",
          subCat:"0").
#mesh08_C04.588.180: mesh08Code(c:"C04.588.180",category:"C", subCat:"04").
```

The object having ***id*** `#neoB_01`, is an instance of the `breast_primarySited_neoplasm` class. Its attributes `descr` and `zone` (which type is `string`) have string values: `"Malignant neoplasm of breast"` and `"Nipple and areola"`, whereas the attribute `site` has value `#A01.236` that is an ***id*** representing an instance of the class `bodyRegion`. Tuples expressing medical knowledge can be declared by using the following syntax:

```
suffer (pat:#21, dis:#neoB_01).
relatedDrug (dis:#C50.9, dru:#L02BG03).
sideEffect (dru:#L02BG03, effect:"Chest pain").
sideEffect (dru:#L02BG03, effect:"Shortness of breath").
classifiedAs (dis:#neoB 01, c:#icd10 C50.0).
classifiedAs (dis:#neoB 01, c:#mesh08 C04.588.180).
```

The tuple of the relation `suffer` asserts that the patient `#p_002` suffers of the disease `#neoB_01`. The same diseases is classified in the ICD10-CM with identifier code `#icd10_C50.0`, and is stored in Mesh tree structure with identifier code `#mesh08_C04.588.180`. By means of the relation `classifiedAs` an user is enabled to querying concepts in the SD-KRF ontology by using one of the identifiers in the original medical ontologies.

4.3 Reasoning over schemas and instances

Integrity constraints, business rules and complex inference rules can be expressed over schemas and instances respectively by means of *axioms* and *reasoning tasks*. For example, the following axiom prevents the prescription of a drug to a patient that has an allergy to a particular constituent of the drug.

```
::-therapyStrategy(patient:P, therapy:T, drug:D),
   hasActivePrinciple(drug:D,constituent:C),
   allergy(patient:P,actPrin:C).
```

Axioms could be, also, used for: (i) specify constraints about transitions behavior. For example, the axiom `"::-P:process(), not start_node(container:P)."` expresses that a `start_node` must exist for each process. Constraints are also used for expressing that a transition links nodes belonging to the same process, and corresponds to an effective edge of the process model as shown in the following:

```
::-transition(from:N1,to:N2), N1:node(container:P1),
   N2:node(container:P2), P1!=P2.
::-transition(from:N1,to:N2), N1:node(start_time:ST1),
   N2:node(start_time:ST2), ST1>=ST2.
::-P:neoplasm_process(), transition(from:N1,to:N2),
   N1:acceptance_node(container:P),
   not N2:anamnesis_node(container:P).
...
```

A reasoning task can be used for expressing a business rule. The following reasoning task, for instance, throws a medical error event when the prescribed dose exceed the recommended dose based on individual characteristics (i.e. age and weight) of the interested patient. Such a check is useful when a `therapeutic_strategy_form` is created while `therapeutic_strategy_definition_node` is active.

```
ID:drugPrescription_medicalError(relativeTo:TASK,timestamp:TIME,msg:MSG):-
    TASK:chemotherapeutic_strategy_form(strategy:STR),
    STR:therapeuticStrategy(patient:P, therapy:T),
    P:patient(bornDate:DATE,weight:W), @age(date,AGE),
    T:therapy(dru:DRUG,dose:DOSE),
    recommendedDose(drug:DRUG, dose:RD, minAge:MA, MinWeight:MW),
    AGE<MA, W<MW, DOSE>RD,
    MSG:="Prescribed dose " + DOSE + "exceed recommend dose " + RD, @newID(ID),
    @now(TIME).
```

The generated prescription error event must be properly handled in the process, for example an error message is visualized by means of a GUI to the physician.

```
ID:view_msg(method:"exception.jar", msg:MSG):-
   X:drugPrescription_medicalError(relativeTo:TASK, timestamp:TIME, msg:MSG),
   @newID(ID).
```

Queries can be also used for exploring clinical processes ontologies in a semantic fashion. For instance `malNeoplasm_f_patient(patient:P)?` returns every female patients suffering of any malignant neoplasm (e.g `P=#21, P=#34` ids are given for answer), where `malNeoplasm_f_patient(patient:P)`:

```
malNeoplasm_f_patient(patient:P):- P:patient(sex:#F),suffer(patient:P,disease:D),
                                   D:malignant_neoplasm().
```

5. Semantic information extraction

In the Static/Dynamic Knowledge Representation Framework classes and instances can be enriched by *concepts descriptors* that are rules allowing to recognize and extract concepts contained in unstructured documents written in natural language. Concepts extracted by descriptors are stored in the knowledge base as instances of the related classes in the ontology.

Considering the example of clinical process described in Section 4 semantic information extraction tasks can be applied to Electronic Medical Records (EMRs), results of examinations, medical reports, etc. coming from different hospital wards in order to populate the clinical process instance related to a given patient.

In the following, a set of semantic extraction rules (i.e. *descriptors*) that allow for extracting patient name, surname, age and disease is shown. Descriptors exploit concepts and relationships referred to the disease, its diagnosis, cares in term of surgical operations and chemotherapies with the associated side effects. Concepts related to persons (patients), body parts and risk causes are also represented. All the concepts related to the cancer come from the ICD10-CM diseases classification system, whereas the chemotherapy drugs taxonomy, is inspired at the Anatomic Therapeutic Chemical (ATC) classification system. Extracted information are exploited to construct, for each cared patient, an instance of lung cancer clinical process.

```
class anatomy ().
    class bodyRegion (bp:string) isa {anatomy}.
        class organ isa {body_part}.
            lung: organ("Lung").
```

```
            <lung> -> <X:token(), matches(X,"[Ll]ung")>.
        ...
    ...
class disease (name:string).
    tumor: disease("Tumor").
    <tumor> -> <X:token(), matches(X,"[Tt]umor")>.
    cancer: disease("Cancer").
    <cancer> -> <X:token(), matches(X,"[Cc]ancer")>.
    ...
relation synonym (d1:disease,d2:disease)
    synonym(cancer,tumor).
    ...
class body_part_desease () isa {disease}.
    lung_cancer: body_part_disease("Lung cancer").
    <lung_cancer> -> <diagnosis_section> CONTAIN <lung> &
                     <X:desease(),synonym(cancer,X)>
    ...
collection class patient_data (){}
    collection class patient_name (name:string){}
        <patient_name(Y)> -> <T:token(),defBy(T, "name:")>
                             <X:token()> {Y := X;} SEPBY <X:space()>.
    collection class patient_surname (surname:string){}
            <patient_surname(Y)> -> <X:hiStr(),matches(X,"sur(:?name)?:")>
                                    <X:token()> {Y:=X;} SEPBY <X:space()>.
        collection class patient_age (age:integer){}
            <patient_age(Y)> -> <X:token(),matches(X,"age:")>
                                <Z:token()>{Y := $str2int(Z);}
                                SEPBY <X:space()>.
    ...
collection class patient_data (name:string, surname:string,
                               age:integer, diagnosis:body_part_disease){}
     <patient_data(X,Y,Z,lung_cancer)> -> <hospitalization_section>
                                          CONTAIN
                                          <P:patient_name(X1)>{X:=X1} &
                                          <P:patient_surname(Y1)>{Y:=Y1} &
                                          <P:patient_age(Z1)>{Z:=Z1} &
                                          <lung_cancer>.
...
```

The classes `diagnosis_section` and `hospitalization_section`, used in the above descriptors, represent text paragraphs containing personal data and diagnosis data recognized by proper descriptors that aren't shown for lack of space. The extraction mechanism can be considered in a WOXM fashion: Write Once eXtract Many, in fact the same descriptors can be used to enable the extraction of metadata related to patient affected by `lung_cancer` in unstructured EMRs that have different arrangement. Moreover, descriptors are obtained by automatic writing methods (as happens, for example, for the cancer and tumor concepts) or by visual composition (as happens for `patient_data`)

Metadata extracted by using the descriptors are stored as class instances into a knowledge base. Using descriptors shown above the extraction process generates the following

`patient_data` class instance for an EMR: `"#1":patient_data("Mario", "Rossi","70",lung_cancer).`

5.1 Implementation issues

A prototype of the SD-KRF has been implemented by combining the JBPM engine JPDL (2011) and the XONTO system Oro et al. (2009a). It is designed to follow a clinical processes life-cycle model based on 3 phases: processes and ontologies *design and implementations*, *execution and monitoring*, *analysis*. The first module exploits the XONTO system. It provides functionalities for importing and/or representing (by using direct "on-screen" drawing and manual specification facilities): ontologies and processes. Moreover, semantic extraction rules (descriptors), that enable the recognition and extraction of information from unstructured sources, can be modeled by exploiting the XONTO approach. A set of business/risk/error rules can be described in terms of ontology constraints and/or reasoning tasks. Acquired and/or represented schemas and instances are stored in a knowledge base and can be queried by using querying and meta-querying capabilities. The *Execution & Monitoring* module is mainly constituted by the JBPM engine that interact with the XONTO system. Process execution is performed in two ways: (i) by using a *workflow enactment* strategy. In this case, a process schema is imported in JBPM and automatically executed involving actors that can be humans or machines (e.g. legacy systems supplying results of medical examinations); (ii) by using a dynamic *workflow composition* strategy. In this case, nodes to execute are selected step by step by choosing the most appropriate one in a given moment. Nodes are chosen by using semantic querying capabilities of XONTO system and executed by JBPM. Queries allows for specifying data and each significant information available in the particular moment of the execution. The execution generates process instances that are stored in a knowledge base. Reasoning, querying and meta-querying over schemas and available instances are possible. This way, process execution can be monitored by running business rules that equip process schemas. Events generated by rules alert process actors that can react for preventing risks and errors. The *Analytics* module aims at allowing analysis of the clinical processes instances after their acquisition. The execution of clinical processes makes available process instances that are also ontology instances. This way a large amount of semantically enriched data becomes available for retrieval, querying and analysis. Analysis are performed by creating reports obtained by semantic query of acquired process instances.

The SD-KRF constitutes an innovative approach for semantic business process management in the field of healthcare information systems. Main features of the presented semantic approach (founded on logic programming) is that it, conversely to already existing systems and approaches, enables to represent process ontologies that can be equipped with expressive business rules. In particular, the proposed framework allows for jointly managing declarative and procedural aspects of domain knowledge and express reasoning tasks that exploit represented knowledge in order to prevent errors and risks that can take place during processes execution. The framework enables, also: (i) manual process execution in which each activity to execute in a given moment is chosen by a human actor on the base of the current configuration of patient and disease parameters, and (ii) automatic execution by means of the enactment of an already designed process schema (e.g. guidelines execution). During process execution, process and ontology instances are acquired and stored in a knowledge base. The system is able to automatically acquire information from electronic unstructured medical documents, exploiting a semantic information extraction approach. Extracted information are stored into a knowledge base as concept instances and are also used to fill XML documents enabling interoperability. The processes execution can be monitored by running (over clinical process schemas and instances) reasoning tasks that implements

business rules. The framework could be used in many application fields by modeling proper domain ontologies and processes.

6. References

Awad, A., Decker, G. & Weske, M. (2008). Efficient compliance checking using bpmn-q and temporal logic, *BPM*, pp. 326–341.

Boley, H., Tabet, S. & Wagner, G. (2001). Design rationale for ruleml: A markup language for semantic web rules, *SWWS*, pp. 381–401.

CCAM (2011). Classification commune des actes médicaux.
URL: *http://www.ccam.sante.fr/*

COIN (2011). Enterprise collaboration and interoperability.
URL: *http://www.coin-ip.eu/*

CORDIS (2011). Community research and development information service.
URL: *http://cordis.europa.eu/*

CPT (2011). Current procedural terminology (cpt): a registered trademark of the american medical association (ama).
URL: *http://www.ama-assn.org/ama/pub/category/3113.html*

Francescomarino, C. D. & Tonella, P. (2008). Crosscutting concern documentation by visual query of business processes, *Business Process Management Workshops*, pp. 18–31.

Georgakopoulos, D., Hornick, M. & Sheth, A. (1995). An overview of workflow management: from process modeling to workflow automation infrastructure, *Distrib. Parallel Databases* 3(2): 119–153.

GLIF (2011). The guideline interchange format.
URL: *http://www.openclinical.org/gmm_glif.html*

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing?, *Int. J. Hum.-Comput. Stud.* 43(5-6): 907–928.

Haller, A., Gaaloul, W. & Marmolowski, M. (2008). Towards an xpdl compliant process ontology, *SERVICES I*, pp. 83–86.

Haller, A., Oren, E. & Kotinurmi, P. (2006). m3po: An ontology to relate choreographies to workflow models, *IEEE SCC*, pp. 19–27.

Hepp, M., Leymann, F., Domingue, J., Wahler, A. & Fensel, D. (2005). Semantic business process management: A vision towards using semantic web services for business process management, *ICEBE*, pp. 535–540.

Hepp, M. & Roman, D. (2007). An ontology framework for semantic business process management, *Wirtschaftsinformatik (1)*, pp. 423–440.

HL7 (2011). Health Level Seven.
URL: *http://www.hl7.org/*

Hornung, T., Koschmider, A. & Oberweis, A. (2007). Rule-based autocompletion of business process models, *CAiSE Forum*.

Hripcsak, G., Ludemann, P., Pryor, T. A., Wigertz, O. B. & Clayton, P. D. (1994). Rationale for the arden syntax, *Comput. Biomed. Res.* 27: 291–324.
URL: *http://dl.acm.org/citation.cfm?id=188778.188784*

ICD (2011). International Classification of Diseases, Tenth Revision, Clinical Modification (ICD-10-CM).
URL: *http://www.cdc.gov/nchs/about/otheract/icd9/icd10cm.htm*

JPDL (2011). jbpm process definition language.
URL: *http://www.jboss.org/jbossjbpm/jpdl/*

LOINC (2011). Logical observation identifiers names and codes.
URL: *http://loinc.org/*

Markovic, I. (2008). Advanced querying and reasoning on business process models, *BIS*, pp. 189–200.

Medeiros, A. K. A. D. & Aalst, W. M. (2009). Process mining towards semantics, *Advances in Web Semantics I*, number 46, pp. 35–80.

MESH (2011). Medical subject headings.
URL: *http://www.nlm.nih.gov/mesh/*

Missikoff, M., Proietti, M. & Smith, F. (2011). Querying semantically enriched business processes, *DEXA (2)*, pp. 294–302.

Montali, M., Torroni, P., Alberti, M., Chesani, F., Gavanelli, M., Lamma, E. & Mello, P. (2008). Verification from declarative specifications using logic programming, *ICLP*, pp. 440–454.

Musen, M. A., Tu, S. W., Das, A. K. & Shahar, Y. (1996). Eon: A component-based approach to automation of protocol-directed therapy., *Journal of the American Medical Informatics Association* 3(6): 367–388.

NCI-EVS (2011). NCI Enterprise Vocabulary Services (EVS).
URL: *http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/vocabulary*

Newell, A. (1980). The knowledge level (presidential address), *AI Magazine* 2(2): 1–20.

OMG (2011). Unified modeling language.
URL: *http://www.uml.org/*

OPCS-4 (2011). The office of population censuses and surveys classification of surgical operations and procedures, 4th revision.
URL: *http://www.openclinical.org/medTermOPCS4.html*

Oro, E. & Ruffolo, M. (2009). Towards a semantic system for managing clinical processes, *ICEIS (2)*, pp. 180–187.

Oro, E., Ruffolo, M. & Saccà, D. (2009a). Ontology-based information extraction from pdf documents with xonto, *International Journal on Artificial Intelligence Tools (IJAIT)* 18(5): 673–695.

Oro, E., Ruffolo, M. & Saccà, D. (2009b). A semantic clinical knowledge representation framework for effective health care risk management, *BIS*, pp. 25–36.

Peleg, M. (2009). Executable knowledge, *Encyclopedia of Database Systems*, pp. 1073–1079.

Peleg, M., Ogunyemi, O., Tu, S., Boxwala, A. A., Zeng, Q., Greenes, R. A. & Shortliffe, E. H. (2001). Using features of arden syntax with object-oriented medical data models for guideline modeling, *in American Medical Informatics Association (AMIA) Annual Symposium, 2001*, pp. 523–527.

PLUG-IT (2011). It-socket knowledge portal - plug-it initiatives.
URL: *http://plug-it.org/*

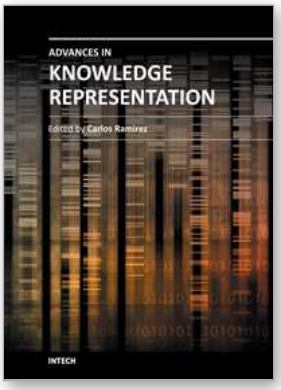
Protégé (2011). Ontology Editor and Knowledge Acquisition System.
URL: *http://protege.stanford.edu*

Pryor, T. A. & Hripcsak, G. (1993). The arden syntax for medical logic modules., *Journal of Clinical Monitoring and Computing* 10(4): 215–224.

Roman, D. & Kifer, M. (2007). Reasoning about the behavior of semantic web services with concurrent transaction logic, *VLDB*, pp. 627–638.

Sackett, D. L., Rosenberg, W. M. C., Gray, M. J. A., Haynes, B. R. & Richardson, S. W. (1996). Evidence based medicine: what it is and what it isn't, *BMJ* 312(7023): 71–72.
URL: *http://bmj.bmjjournals.com/cgi/content/full/312/7023/71*

Shortliffe, E. H. (1976). *Computer-Based Medical Consultations: MYCIN*, Elsevier/North-Holland.
SNOMED (2011). Systematized nomenclature of medicine-clinical terms (snomed ct). URL: *http://www.ihtsdo.org/snomed-ct/*
Stojanovic, L. & Happel, H.-J. (2006). Ontoprocess – a prototype for semantic business process verification using swrl rules, *ESWC*.
SUPER (2011). Semantic utilised for process management within and between enterprises. URL: *http://www.ip-super.org/*
Sutton, D. R. & Fox, J. (2003). The syntax and semantics of the proforma guideline modeling language., *JAMIA* 10(5): 433–443.
ter Hofstede, A. H. M., van der Aalst, W. M. P., Adams, M. & Russell, N. (eds) (2010). *Modern Business Process Automation - YAWL and its Support Environment*, Springer.
UMLS (2011). Unified medical Language System. URL: *http://www.nlm.nih.gov/research/umls/*
van der Aalst, W. M. P. (1999). Formalization and verification of event-driven process chains, *Information & Software Technology* 41(10): 639–650.
van der Aalst, W. M. P. (2009). Business process management, *Encyclopedia of Database Systems*, pp. 289–293.
Weske, M. (2007). *Business Process Management: Concepts, Languages, Architectures*, Springer.
Wetzstein, B., Ma, Z., Filipowska, A., Kaczmarek, M., Bhiri, S., Losada, S., Lopez-Cob, J.-M. & Cicurel, L. (2007). Semantic business process management: A lifecycle based requirements analysis, *SBPM*.
White, S. (2006). Business process modeling notation specification.
WHO (2011). World health organization. URL: *http://www.who.int/classifications/en/*
Wohed, P., van der Aalst, W. M. P., Dumas, M., ter Hofstede, A. H. M. & Russell, N. (2006). On the suitability of bpmn for business process modelling, *Business Process Management*, pp. 161–176.
XPDL (2011). Xpdl2.1 complete specification. URL: *http://www.wfmc.org/xpdl.html*



Advances in Knowledge Representation

Edited by Dr. Carlos Ramirez

ISBN 978-953-51-0597-8
Hard cover, 272 pages
Publisher InTech
Published online 09, May, 2012
Published in print edition May, 2012

Advances in Knowledge Representation offers a compilation of state of the art research works on topics such as concept theory, positive relational algebra and k-relations, structured, visual and ontological models of knowledge representation, as well as detailed descriptions of applications to various domains, such as semantic representation and extraction, intelligent information retrieval, program proof checking, complex planning, and data preparation for knowledge modelling, and a extensive bibliography. It is a valuable contribution to the advancement of the field. The expected readers are advanced students and researchers on the knowledge representation field and related areas; it may also help to computer oriented practitioners of diverse fields looking for ideas on how to develop a knowledge-based application.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821