CONFlexFlow: Integrating Flexible Clinical Pathways into Clinical Decision Support Systems using Context and Rules

Wen Yao
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802 USA
Ad howxy119@psu.edu

Akhil Kumar
Smeal College of Business
The Pennsylvania State University
University Park, PA 16802 USA
akhil@psu.edu

ABSTRACT

We propose **CONFlexFlow (Clinical cONtext based Flexible workFlow)** as a novel approach for integrating clinical pathways into Clinical Decision Support Systems (CDSS). It recognizes that clinical pathways involve frequent deviations and require considerable flexibility. Thus, integration of flexible pathways is critical for the success of CDSS. Further, it is based on a better understanding of clinical context through ontologies, and bringing them to bear in deciding the right rules for a certain activity. We also describe an approach for dynamically realizing context dependent medical activities in a clinical pathway based on the needs of a specific case. To illustrate the feasibility of our approach, we propose an implementation framework and present a proof of concept prototype using multiple open source tools. The role of semantic web technologies in realizing flexible clinical pathways and integrating them into CDSS is highlighted. Preliminary results from our initial implementation are discussed.

Keywords

CDSS, clinical pathway, rules, context, ontologies, ad hoc subprocess.

1. INTRODUCTION

Workflows play an important role in clinical environments by delineating the steps through which the treatment of a patient progresses. The temporal order and correct coordination of the various steps are clearly important. The workflows in these environments are called *clinical pathways*. These pathways generally follow well-established standards or clinical guidelines. However, they differ from other workflows found in business and production environments because clinical processes involve frequent deviations, and hence there is a need for considerable flexibility. Typically, a medical facility develops clinical pathways from clinical guidelines on the basis of its local resources and settings. Moreover, the pathway is further customized into a treatment scheme to suit an individual patient's needs [8].

Clinical workflows are highly dynamic, context sensitive, event driven, and knowledge intensive. In this respect, they are quite unique. In general, a patient interacts with a Primary Care Physician's (PCP)

office, a pharmacy, labs, and one or more specialists, etc. In this setting, it is important to maintain coordination and flow of information among these various entities to ensure an optimal outcome. The need for new modeling techniques for designing such flexible workflows is motivated by several considerations. First, although many research efforts are geared towards establishing international healthcare standards (e.g., HL7 [23]) and a representation for sharable guidelines (e.g., GLIF [37]) for clinical practice, formal models of executable and flexible clinical workflows are very few (e.g., [16, 55]). The execution of a clinical workflow is highly dependent on the existing body of medical knowledge, available resources, and specific case data. For example, doctors with different skill levels and fields of expertise may offer differing treatments to the same patient. A sudden rise in a patient's blood pressure may require an additional test and alter her treatment in the subsequent pathway. Thus, different pathways can arise based on case specifics and the proclivities of attending doctors, and it is important to formally model these scenarios. Second, since medical staff handles a lot of cases each day, they are prone to making mistakes in prescribing medications, performing procedures, and even making diagnoses [11, 30, 51]. Hence, a Clinical Decision Support Systems (CDSS) and Computer Interpretable Guidelines (CIGs) [49] can assist care professionals in reducing likelihood of errors and improving care quality.

Our goal is to show how flexible clinical pathways can be designed taking into account medical knowledge in the form of rules, and also detailed contextual information for a medical workflow involving multiple participants to improve care quality. We propose a methodology for designing formal workflow models that capture medical knowledge and context in a common framework, and yet allow flexibility. Since a clinical workflow should naturally be aligned with clinical guidelines, it is necessary to ensure that it is formal and correct so that integrating decision support into this workflow can be helpful. The methodology is based on a formal rule and context taxonomy using ontologies. The rule taxonomy organizes the rules into a hierarchy while context encompasses aspects of patients, providers, resources, and environment. We will focus on how context is captured, described and summarized, and how rules are developed. A proof of concept prototype is built and preliminary results are given to show the feasibility of our approach. In this paper, the terms pathway and workflow are used interchangeably.

The organization of this paper is as follows. Section 2 provides background and related work. Then we introduce a meta-model for a CDSS, and present the architecture for system implementation in Section 3. Next, Section 4 presents an ontology-based context model and discusses rule-based medical reasoning. Based on this knowledge framework, we present our approach for designing flexible clinical workflows using BPMN 2.0 ad hoc subprocesses and describe a preliminary implementation in Section 5. Later, Section 6 gives a brief discussion and plans for future work, followed by a conclusion in Section 7.

2. BACKGROUND AND RELATED WORK

A Clinical Decision Support System (CDSS) is an interactive computer software designed to assist health professionals with decision making tasks, such as preventing adverse drug events at the point of care [51]. Although many methodologies are employed in designing CDSS, including Bayesian networks, neural networks, and genetic algorithms, we focus on rule-based approaches in this study. Rule-based CDSS, derived from expert systems research [11], are knowledge based systems that integrate a medical knowledge base, patient data, and an inference engine to generate case specific advice. An overview of the state of the art of rule-based CDSS is given in [11, 43, 49].

Although CDSS are said to be very helpful to assist in the clinicians' work, still they are reluctant to adopt CDSS because of unnecessary workflow disruptions [27]. The importance of integrating a CDSS with a clinical workflow has been stressed by other studies as well [19, 27, 52]. Kawamoto et al. [28] note that CDSS interventions that are presented automatically and fit into a workflow of medical staff are more likely to be adopted, and a CDSS that makes recommendations is better than one that only gives an assessment. Another challenge is to ensure the correctness of decision support by following standard guidelines. The objective of a CDSS is to deliver the "*right knowledge* to the *right people* in the right form *at the right time*" [48]. Thus, integrating medical guidelines along with clinical data into a CDSS aims to deliver evidence-based recommendations to care providers at various points of care. Table 1 summarizes a variety of approaches from different research communities and compares them in terms of their knowledge base organization and workflow integration capability. Further details of these studies are discussed next along with the role of ontology in healthcare.

Table 1. Comparison of existing approaches for modeling clinical processes

Approach \ Criteria		Knowledge/data source			Workflow Integration	
		Form of clinical data	Clinical guidelines	Formal vocabulary	Workflow flexibility	Workflow patterns
Process modeling approach	Workflow modeling [7, 9, 22, 34, 47]	unstructured	---	---	++	++
	Ontology modeling [8, 13, 16]	ontology	+	---	++	+
Software engineering approach	Little-JIL [14, 15, 40]	EMR	++	---	++	+
	Model integrated approach [32, 33]	EMR	++	---	+	+
Guideline modeling approach	CIGs [19, 23, 39, 44, 45]	N/A	++	N/A	N/A	N/A
	Ontology-based approach [6, 55]	Unstructured, EMR	++	---	++	+

Notation: N/A: not applied; --- not supported/considered; +: weakly supported, ++strongly supported

2.1. Clinical Workflow

The business process management community has devoted a lot of effort in designing customized and flexible clinical processes using formal workflow languages (e.g., BPMN and BPEL) that are executable by existing workflow engines. Lenz and Reichert [31] conducted a survey of IT support for healthcare processes and emphasized the importance of flexible workflow design in supporting clinical decisions. Such systems have been developed to enable dynamic changes in predefined process models, such as $ADEPT_{flex}$ [47]and AgentWork [34]. Research on context-aware workflow design [7, 9, 22] also belongs to this area, but focuses more on the integration of context in binding specific services or constructing subprocesses, whereas medical knowledge is deemphasized. These techniques are useful and highlight the criticality of context in designing a flexible workflow.

However, there has also been work from the software engineering community on modeling medical processes. In particular, limitations of current workflow languages have been observed [56], and an alternative approach based on Little-JIL language [53] has been proposed. Little JIL is a language that centers on a coordination diagram of the process that described bya hierarchical task decomposition. It helps to coordinate agents and their activities, and allows steps to be performed in sequence or parallel. A Little-JIL description can be verified using finite state machine verification techniques. Approaches for formally defining and analyzing medical processes using Little JIL are discussed in [14, 15, 40]. This stream of research is useful for its focus on improving patient safety and detecting medical errors, but it

does not emphasize flexibility or decision support. Mathe et al. [33] developed the Model-Integrated Clinical Information System (MICIS) using model-based design techniques to represent complex clinical workflows in a service-oriented architecture. Thus, treatment protocols are transformed into executable constructs such as Web services or BPEL processes to promote software reuse and maintainability.

2.2. Clinical Guidelines

The medical informatics community models clinical processes as guidelines or care plans. Their focus is on medical decision making by interpreting situations based on best practice. They also formalize process definitions in a way that reflects clinical tasks and constraints as clinicians perceive them.

Medical guidelines were originally expressed as free-format text documents to assist medical decision-making during diagnosis, management and treatment within different areas of healthcare. Recently, various approaches have been proposed to represent Computer-Interpretable Guidelines (CIGs), such as Arden Syntax, Asbru, EON, GLIF, GUIDE, PRESTIGE, PRODIGY, PRO*forma*, and SAGE (see [19, 23, 37, 39, 44, 45]). CIGs can produce personalized recommendations during patient encounters and reduce variance in patient treatment. Peleg et al. [45] reviewed six CIG modeling approaches and established a consensus on their common structure. They represent clinical guidelines as *plans*, whose components represent decisions, actions, and the relationships among them. *Decision steps* are used for conditional and unconditional routing of the flow, while *action steps* are used to specify a set of tasks or a sub-plan to be carried out. A review of systems using CIGs was conducted in [25]. The main disadvantage is that all CIGs require an execution engine that is not freely available. Complex workflow patterns cannot be easily modeled in this way, and validation (e.g., checking for possible deadlocks) of such workflows is difficult.

2.3. Ontology in Healthcare

An ontology is a formal specification of the concepts within a domain and their interrelationships. The Web Ontology Language (OWL) [10] is a semantic markup language. It is becoming a standard for sharing ontologies among people and also software agents on the web. Since OWL uses first-order logic,

the models, and description of data in these models, can be formally verified. Thus, inconsistencies in the model can be detected, and new information can also be inferred, by machine reasoning.

The use of ontology in healthcare has spanned a variety of research areas including domain conceptualization, clinical context modeling, and workflow modeling. The biomedical community has invested much effort towards building domain ontologies, such as UMLS [12] and GALEN [46], which focus on various medical disciplines and also allow medical terminologies to be shared. Researchers from the computer and information science community have used ontology to build context models in healthcare to generate context-aware alerts and reminders [26, 29, 41]. These approaches are applied in pervasive environments where embedded devices are available and used for collecting medical data. An *ontology-based context model* provides interoperability among heterogeneous devices and systems, and allows context reasoning based on user-defined rules.

Recently, ontology-based approaches have been used to enhance the semantics of clinical workflows and make customizations accordingly [8, 13, 16, 55]. For example, Dang et al. [16] built an ontological knowledge framework to represent important entities (such as roles and resources) in healthcare by capturing the context in clinical pathways. Based on this framework, they developed a system [17] that enables the creation and execution of personalized medical workflows using BPEL language. Similar studies include SEMPATH [8] and KON^3 [13] but they focus more on rule-based reasoning. Ye et al. [55] proposed a clinical pathway ontology to represent and exchange pathway-related knowledge. They presented clinical pathways as interconnected hierarchical models including the top-level outcome flow and intervention workflow level along a care timeline with the assistance of semantic rules for modeling temporal knowledge. Compared to these approaches, our clinical processes are well structured in general; however, some complex groups of medical activities are only loosely-defined at design time. This allows a customized pathway to be dynamically generated on the fly at runtime. Further, we argue that a BPMN model is more understandable by care professionals since it provides a graphical representation. It also has rich semantics for modeling medical tasks and thus is suitable for modeling clinical processes.

Next, we discuss our framework.

3. AN INTEGRATED FRAMEWORK–CONFlexFlow

We propose a **Clinical CONtext based Flexible WorkFlow (CONFlexFlow)** approach for designing the medical knowledge base and providing decision support. Our study complements earlier research studies described in the related work above. We aim to bridge the gap among research from the medical informatics, business process management and the semantic web communities. Our main contributions are to: (1) develop an *integrated ontology model* to capture contextual knowledge that has an impact on clinical practice; (2) use semantic rules to encode medical procedural knowledge from clinical guidelines; (3) model *clinical processes* using *a standard workflow language (BPMN 2.0)* and make them *adaptable* to the continuously changing context; (4) present a system architecture for implementing CONFlexFlow and develop a proof of concept prototype using multiple open source tools.

3.1. A Running Example

Figure 1 represents a simplified BPMN clinical workflow created in the web-based Signavio-Oryx editor [3]. BPMN 2.0 [38] can support various types of tasks and workflow patterns. It is executable and supported by existing workflow engines. In this diagram, *pools* (represented by rectangular blocks) correspond to workflow participants, including patient, CDSS, PCP office, pharmacy, and lab. A message flow (shown by dotted lines) is used to coordinate communication between two participants. *Lanes* are used to organize and categorize activities within a pool, say between a doctor and a nurse. The control flow among activities within a pool is shown by solid lines. This pathway shows that a patient having any symptom visits the PCP office and is examined by the care providers. The doctor writes a prescription (Rx) and sends it to the pharmacy. The pharmacy prepares and dispenses the medicine, and the lab is responsible for various tests. Activities or tasks are shown in round rectangles, while events are in circles. Complex gateways (shown in diamonds with an asterisk) allow one or more outgoing branches based on the results of conditions. *Parallel gateways* (shown in a diamond symbol with an asterisk) create parallel paths without checking any conditions. This is a formal description of the coordination among activities in this process.

In this diagram, “Prepare Rx” is a *reusable subprocess* whose execution semantics are strictly and explicitly defined, while “Do tests” is an *ad hoc subprocess* that is realized from other isolated activities. It is “loosely defined” at design time because its instantiation is *dynamically* determined by the outcome of previous activities. For example, if a doctor upon examination suspects that a patient may have suffered a heart attack, then “EKG” and “ECG” would be executed in the subsequent pathway. In contrast, a “Blood test” might be needed if a patient is suspected of hypertension. Thus, using *a strictly predefined subprocess* to design *an uncertain and dynamic activity* is not practical since it lacks *flexibility and adaptability*. *Context is needed to provide execution semantics for the "loosely-defined" regions of such clinical processes at runtime*.

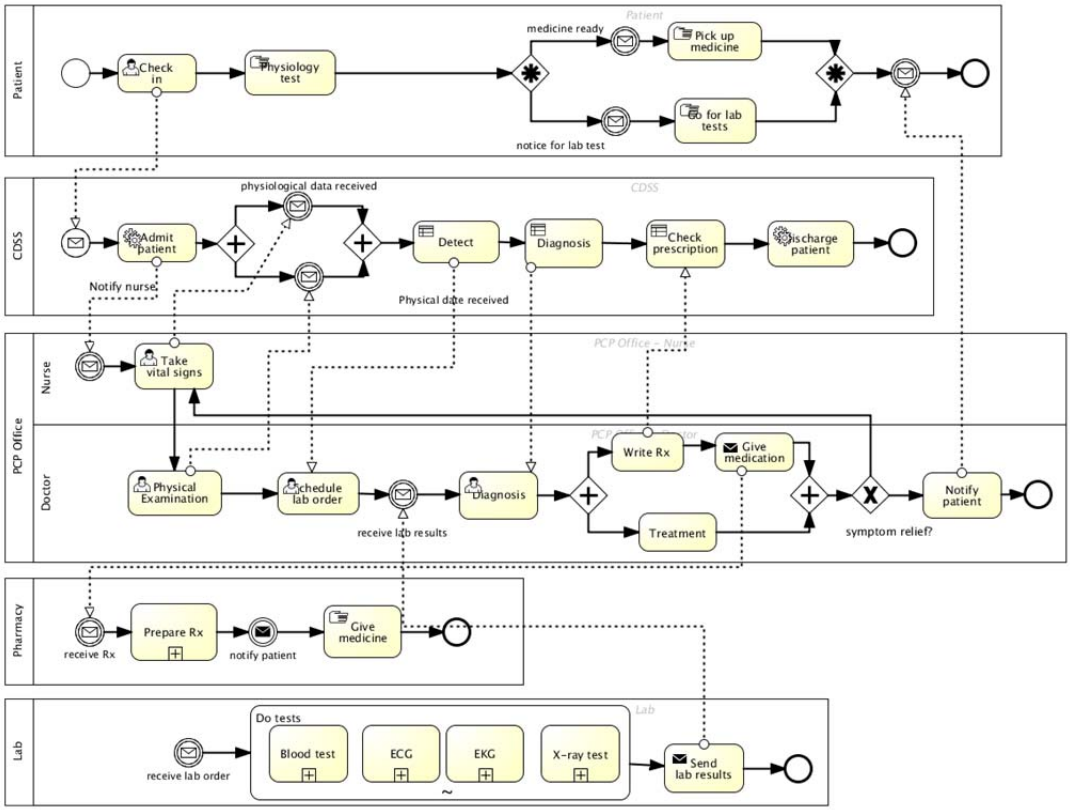


Figure 1. A simplified BPMN clinical workflow modeled in Signavio-oryx editor [3]

When a generic *process model* like the one in Figure 1 is actually executed for a specific patient, it is called a *process instance*. Thus, the process instance for patient 'Sue' with id 'P1001' is different from that for patient 'Jack' with id 'P1003', and so on.

3.2. A Meta-model

Figure 2 represents a meta-model that describes the semantic relationships among key concepts used in CONFlexFlow. It also introduces some of the terminology used in this paper.

Clinical workflows are composed of *activities* performed by *roles* or *participants*. *Rules* are derived from *clinical guidelines* and use *medical ontologies* for clinical reasoning. Semantically related rules are categorized into *rule groups*. A rule group can be associated with one or more workflow activities and it is automatically triggered when the activity node is reached. The data required to evaluate the rule condition is modeled as a *context*, which captures all entities that have an impact on the clinical practice. It has subclasses, *low-* and *high-level context*. Low-level context is explicit and is directly collected from user inputs or other through data sources like medical sensors. Examples of such context are are: patient name, physical location, body temperature, etc. They are used to deduce high-level, implicit context, e.g. about patient diagnoses, by rule-based reasoning. Rules are triggered by contextual data and produce actions, which are generally in the form of *reminders*, *alerts*, and *recommendations*. They can provide evidence-based suggestions for medical decision making within atomic and composite activities.

We employ a variety of BPMN 2.0 elements (see Figure 3) for modeling a flexible clinical workflow. An *atomic activity* is indivisible and is used to compose *processes* and *composite activities*. There are various types of atomic activities. For example, a *human task* requires user interaction with the assistance of a software application while a *manual task* is expected to be performed without the aid of any application. A *service task* is done automatically by a Web service, while a *rule task* handles complicated business logic. They are used for modeling different types of medical tasks. *Composite activities* include *embedded* and *reusable subprocesses*. In this study, we mainly use *ad hoc subprocesses*, which are "loosely defined" at design time. Then at runtime a more specific (or "concrete") description is realized based on the actual context. The "Do tests" activity in Figure 1 is an example of such a process because we do not know at design time what tests will be prescribed by the doctor. In contrast, a *structured process* is strictly defined and the execution semantics are the same for all its running process instances. More details about this notation can be found in the BPMN 2.0 specification [38].

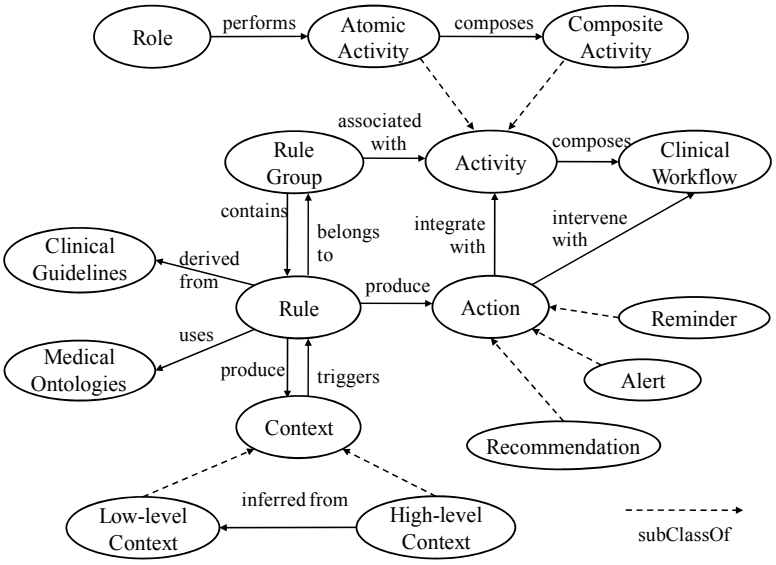


Figure 2. A meta-model for CONFlexFlow

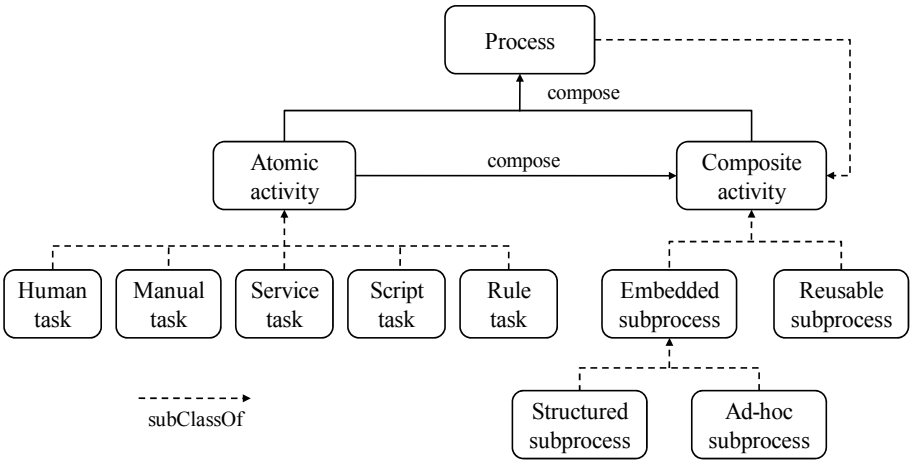


Figure 3. A hierarchy of BPMN 2.0 activity notations [38] for modeling clinical processes

3.3. System Architecture

Figure 4 depicts the system architecture of CONFlexFlow based on our meta-model. Several roles are involved in this system. Knowledge engineers communicate with domain experts and collect medical knowledge. Other medical resources and documents published by health organizations can be consulted as well for this purpose. Then they use ontology and rule editors to build medical ontologies and rules to formalize clinical guidelines. A *Context ontology* is developed to capture clinical context acquired from a variety of distributed sources, such as laboratory information systems, surgical information systems,

sensors and workflow engines. Then the *rule engine* triggers rules, evaluates conditions and takes actions accordingly. The produced actions are tightly integrated into running clinical process instances. Various tools used in implementing several of the modules in this architecture are shown in the figure.

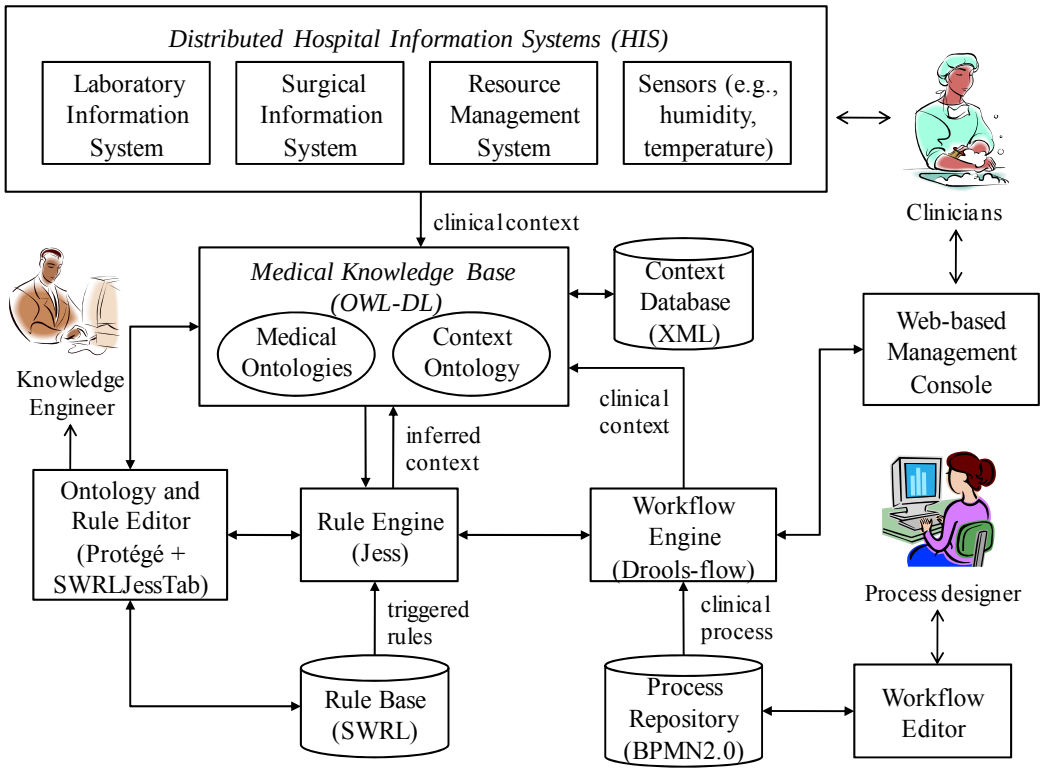


Figure 4. System architecture of CONFlexFlow

Meanwhile, process designers use workflow design tools to create clinical process models and store them in the process repository. They can be initialized and executed by the *workflow engine* to automate patient flow. At runtime, a process instance is executed and fed with specific case data (e.g., patient body temperature). Workflow participants, e.g. various care professionals, manage clinical workflows through a web-based management console to perform clinical activities and monitor their current process instances. They are presented with reminders, alerts, and recommendations based on the clinical reasoning carried out in the rule engine. Furthermore, the clinical workflow is adapted based on the execution semantics generated from the current context of an individual patient. Every activity may produce additional case data (e.g., patient symptoms and lab test results) and update the context continuously.

Thus, this architecture captures the nexus of workflow, rules and context. It is important to note that *a clinician will still be in control*. The pathways and alerts generated by the CDSS can be overridden and adjusted per users' stated preferences.

In the following sections, we describe the critical components of our framework in more detail.

4. CLINICAL KNOWLEDGE REPRESENTATION AND SEMANTIC RULES

The knowledge used for medical decision making in patient encounters includes shared understanding of medical domain concepts, contextual data that characterize a specific clinical process and the procedural knowledge encapsulated in rules. In this section, we discuss the representation of and reasoning with clinical knowledge in CONFlexFlow.

4.1. Ontological Knowledge Model

As discussed above, ontologies are explicit formal specifications of terminologies in a domain and the relationships among them. They can facilitate knowledge sharing, logical inference and knowledge reuse. Ontologies are widely accepted as a useful instrument for modeling context; thus, we use them to model clinical context that can be obtained from heterogeneous sources and represented in different formats. Further, domain ontologies from medical disciplines are included in our knowledge framework since they are critical for decision support. We use Web Ontology Language-Description Logic (OWL-DL) [10] to encode our model using Protégé 3.4 [50], which is a popular tool for ontology editing and representation. This model can be shared across a healthcare network for collaboration.

4.1.1. Clinical Context Ontology

According to Dey [18], *context is any information that is used to characterize the situation of an entity*. Moreover, a system is context-aware if it uses context to bring to bear in the provision of relevant information and/or services, where relevance is naturally situation dependent. In a clinical workflow, we define *context* as any information that can be used to influence medical decision making or clinical workflow routing. For example, a patient's medical history might affect a doctor's prescription, and a patient's lab test result could likely affect her subsequent treatment. A normal test result would lead to a

"follow up check," while an abnormal result may necessitate customized treatment along a new clinical pathway. Clearly, proper representation of context plays a critical role in the CDSS. Here, we formally model the clinical context using an ontology-based approach by capturing the important concepts in the clinical process. At a preliminary level, we consider the following categories of context:

- *Patient-related context* includes medical history, age, gender, physiology data (e.g., blood pressure, body temperature, and heart rate), X-ray test results, mobility state, etc. Patient data can be obtained from human input, medical devices (e.g., physiological monitors), and a central EMR repository.
- *Clinical staff-related context* concerns information related to doctors, nurses, lab staff, pharmacists and other providers. They are the major participants in the clinical workflow and their decision can affect the patient treatment and the final outcome. Their attributes include expertise area, level of expertise, gender, availability, workload, desired alert level, etc.
- *Resource-related context* concerns the information about resources (e.g., devices) that are needed for workflow execution. This might include a defibrillator's availability and current location, the number of available wheel chairs, stock levels of medicines, etc. Additional information is required about the environment at a particular *facility* such as a surgery room's humidity, temperature, and schedule.
- *Location-related context* concerns the current location of persons, devices, and other movable assets. This information is captured by RFID devices, smart phones, or through other technologies. it is helpful in dealing with emergencies, such as, say, the need for an OXRB bottle in surgery.

Figure 5 shows a partial representation of our ontology-based clinical context model that describes the main entities of interest, their properties and semantic relationships. Example entities include hospital staff, patients, resources, facility, etc. Their relationships are also described: e.g., "*locatedIn*" between *Person* and *Location* to denote that a person has a location. The pair of relations "*treatedBy*" and "*treats*" between *Patient* and *Staff* help to manage the medical staff needed for patient treatment.

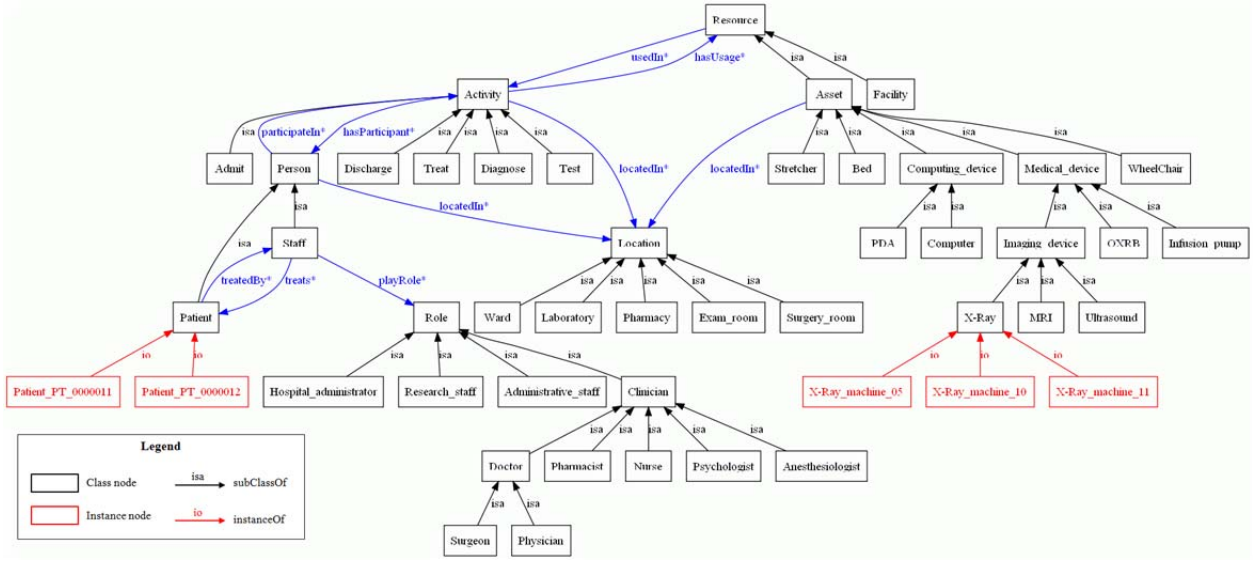


Figure 5. Partial representation of the clinical context ontology

We encoded this context ontology in OWL-DL [10], which is a sublanguage of OWL and has become a standard ontology language in the semantic web community. OWL-DL strikes a good balance between computational and expressive power requirements. It follows an object-oriented approach to describe the structure of a domain in terms of classes, their properties and semantic relationships. It offers inference capabilities through the OWL properties like inversion, symmetry and transitivity.

We used an open source platform Protégé 3.4 [50] to create our ontology since it provides a graphical interface to model ontologies in OWL-DL. Our model was based on an existing hospital ontology which is now available on the web [54]. As shown in Figure 6, six individuals are created as instances of the class *Patient*. The *asserted properties* are low-level context directly entered by the user or fed by other services. For example, the ID, name, gender and vital signs can be entered by the nurse, while physical location can be detected by the RFID tracking service. This patient shows symptoms of chest pain, such as coughing, dyspnea, fatigue, and nausea, which are pre-defined in the medical ontologies and available for nurses to select. Similarly, the other asserted properties show the current activities of this patient and attending clinicians, which will likely change as this process proceeds. The *inferred properties* are high-level context deduced from ontology- and rule-based reasoning. For example, a hypothetical diagnosis,

such as chronic heart failure, is inferred from these symptoms according to practice guidelines. Our CDSS makes recommendations about patient diagnoses and treatments based on medical guidelines as well as the clinical data. The details of rule-based reasoning are discussed Section 4.2.

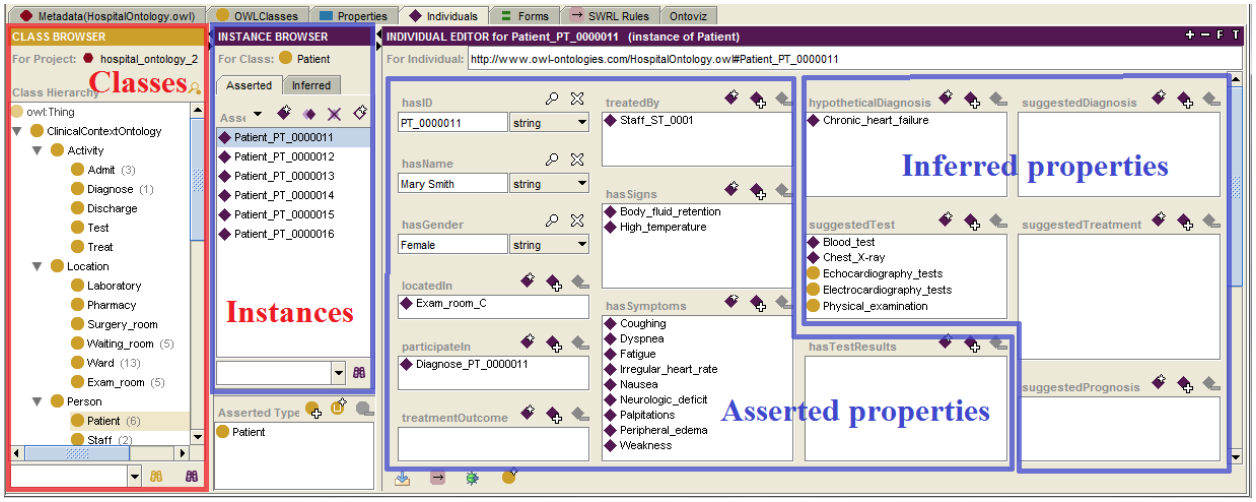


Figure 6. Protégé user interface for editing OWL-based ontology

4.1.2. Medical Domain Ontologies

Medical domain ontologies are created to provide *a shared understanding of concepts in specialized medical disciplines*. Thus, they provide important and formal vocabularies for a CDSS. In this study, it is impossible to include every specialty in the medical domain. Hence, we adopted the already published heart failure ontology named HF_ontology [4] in our prototype. This ontology was developed by the HEARTFAID team at University of Calabria in accordance with the guidelines of the European Society of Cardiology. It is useful for medical decision making related to heart disease.

We use the HF_ontology to infer patient conditions, diagnosis and treatment, as well as the level of severity. Figure 7 shows a partial hierarchy of the HF_ontology. Important concepts such as *patient characteristics*, *tests*, and *treatments* are formally modeled and their relationships are represented as well. *Treatment* includes *medication*, *device therapy*, *surgical therapy*, *patient education*, etc. *Medication* can further be categorized into *ACE inhibitor*, *Angiotensin receptor blocker*, *Beta blocker*, *Digoxin*, *Diuretics*, and *Spironolactone*. This ontology is relatively stable and is updated when there is any change in the shared understanding of heart disease.

The concepts from the HF_ontology are integrated with our context ontology in Figure 5. For example, patients have signs, symptoms, suggested tests, suggested treatment, etc. These constraints or attributes are captured by object properties, such as *hasSigns*, *hasSymptoms*, *suggestedTest*, *suggestedTreatment*, etc., accordingly between class *Patient* in the context ontology and classes *Signs*, *Symptoms*, *Tests* and *Treatment* in the HF_ontology. This model is implemented using Protégé 3.4 and encoded in OWL-DL. Although we only consider the heart failure ontology in this study, domain ontologies for other diseases can be easily incorporated in future.

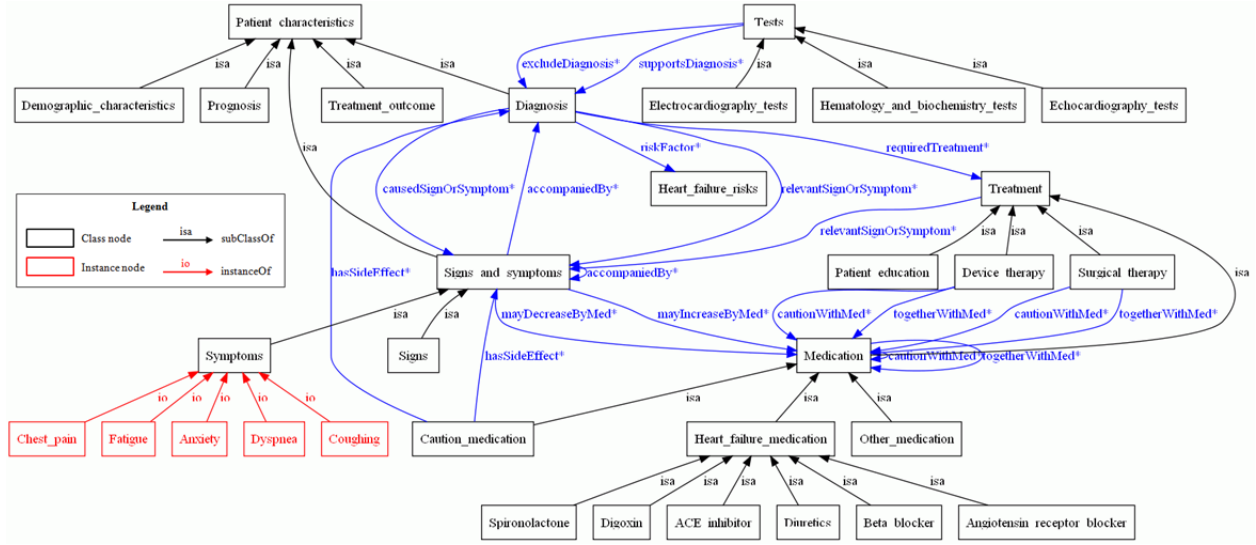


Figure 7. A partial representation of the heart failure ontology (adopted from [4])

4.2. Semantic Rules for Clinical Reasoning

Rules embody medical procedural knowledge and are used to help make complex clinical decisions through logical reasoning, often in real time and in response to critical events. They can also handle exceptional situations. This section discusses how semantic rules are represented and rule-based reasoning is performed. We also discuss their integration into clinical workflows.

4.2.1. Medical Rules

A clinical workflow may have thousands of rules for a variety of purposes (e.g., patient diagnosis, treatment) that are applicable in different areas of medicine (e.g., heart failure, diabetes, pediatrics, etc.). But only a small part of the rule set is triggered for a specific activity within a process instance. Managing

and integrating medical knowledge in the form of rules and applying results from rule-based reasoning into a clinical pathway is important in a CDSS. We organize rules into *rule categories* according to the stage at which they are applied in the clinical process. To obtain these rules we consulted the American Heart Association (AHA) [2] and the American Academy of Family Physicians (AAFP) [1].

Figure 8 shows a hierarchy of our rules categorized into *patient evaluation*, *diagnosis*, *treatment* and *prescription checking*. Each rule category is associated with a medical activity in a clinical pathway and contains one or more rules that represent relevant knowledge. It is further divided into sub-categories. For instance, "prescription checking" is subdivided into "allergy checking," "drug interaction checking," "dose checking," and "insurance checking." So, if a prescribed drug interacts with a patient's medication, the system can propose an alternative to the doctor for approval. New rules can also be developed and added in this hierarchy. Thus, Figure 8 serves as a guide for organizing rules systematically.

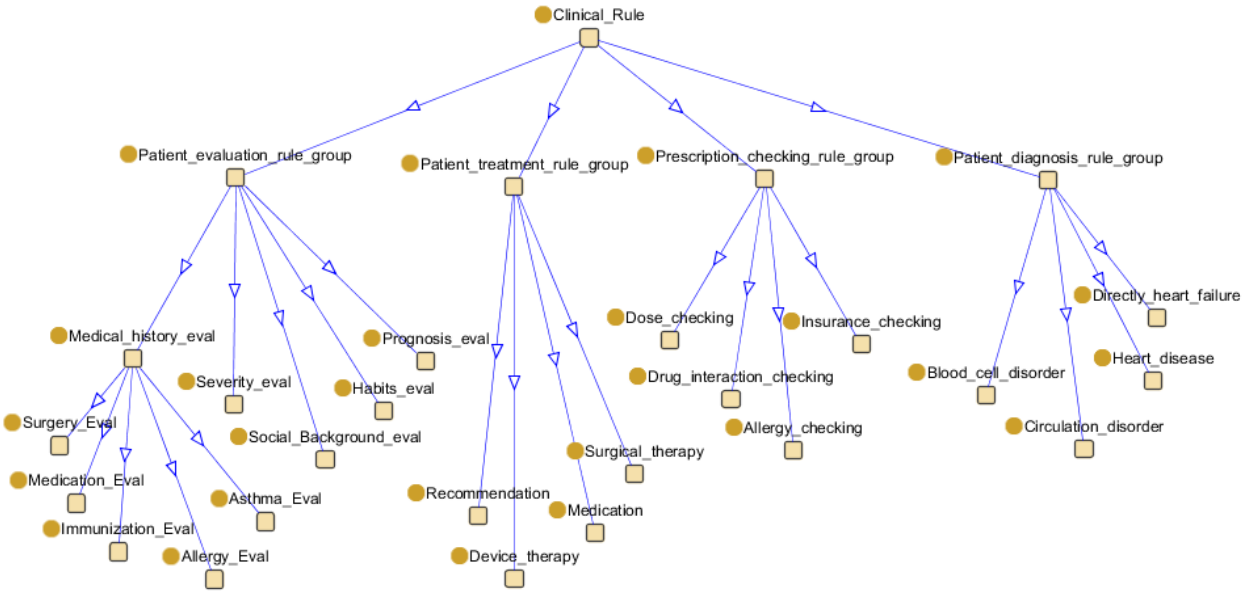


Figure 8. A semantic hierarchy of rules based on clinical knowledge

4.2.2. Semantic Rules in SWRL

We use Semantic Web Rule Language (SWRL) [24] to encode rules for user-defined reasoning. SWRL is the standard rule language for the semantic web based on OWL and RuleML. SWRL utilizes the typical logic expression "antecedent → consequent" to represent semantic rules. Both the antecedent (rule body)

and the consequent (rule head) can be conjunctions of one or more atoms written as "$atom_1 \wedge atom_2 \ldots \wedge atom_n$." If all the atoms in the antecedent are true, then the consequent must also be true. Note that the symbol ∧ means conjunction; ?x stands for a variable (i.e., an instance or individual); and → means implication. The consequent part of a triggered rule can be used to update the context database, issue reminders and alerts, and more importantly, customize an ad hoc subprocess.

Table 2 shows several example rules from different rule categories and their triggering conditions. The information embodied in these rules is provided to care professionals as recommendations to avoid deviation from clinical guidelines. *Patient Evaluation Rules* (PER) evaluate a patient's medical history, social background, habits, symptoms, etc. prior to a physical examination; thus, a hypothetical diagnosis can be presented to a clinician during patient examination. *Patient Diagnosis Rules* (PDR) evaluate a patient's signs, lab test results, other relevant medications, and provide recommendations for diagnosis decisions, such as systolic or hypertensive heart failure. Although signs and symptoms describe the same conditions, they are essentially different.

Signs are what a doctor observes (e.g., high blood pressure and abnormal heart rhythm), while *symptoms* are what a patient experiences (e.g., fatigue and dyspnea). Symptoms can characterize a disease and reported by a patient to a nurse; signs are usually detected and logged by a doctor during examination. *Patient Treatment Rules* (PTR) provide suggestions for treatments such as medications, therapy or patient education, according to the confirmed diagnosis. *Prescription Checking Rules* (PCR) deal with drug interactions (e.g., Carbidopa and Levodopa), dosage checking and allergy-drug effects to avoid prescription errors. For example, rule PCR1 will generate an alert message if a patient who is allergic to Aspirin is given any medication that contains Aspirin. In addition, there are other rules that concern resources such as staff, and pertain to their schedules, availability, etc.

4.2.3. Semantic Reasoning using Jess Rule Engine

The execution of SWRL rules requires the availability of a rule engine that performs reasoning using a set of rules and facts as input. Any new facts that are inferred are used as input to potentially fire more rules

(i.e., forward chaining). In our implementation, we use the Jess rule engine [21] to enable SWRL reasoning on the Protégé data set since it is small, light, and one of the fastest available. The Jess rule engine is implemented in Java and supports forward and backward chaining. It is stable, well supported, and has been successfully used by others.

Table 2. Example user-defined rules (medical knowledge acquired from AAFP [1] and AHA [2])

Category/ Intervention	Scenario	Reasoning rules represented in SWRL
Patient Evaluation Rules (PER) - "Physical Examination"	Chronic heart failure detection	PER1: Patient(?p) ∧ hasSymptoms(?p, Dyspnea) ∧ hasSymptoms(?p, Fatigue) ∧ hasSymptoms(?p, Peripheral_edema) ∧ hasSymptoms(?p, Palpitations) ∧ hasSymptoms(?p, Coughing) ∧ hasSymptoms(?p, Nausea) ∧ hasSymptoms(?p, Neurologic_deficit) → hypotheticalDiagnosis(?p, Chronic_heart_failure)
	Suggested tests in presence of heart failure	PER2:Patient(?p) ∧ hypotheticalDiagnosis(?p, Chronic_heart_failure) → suggestedTest(?p, Physical_examination) ∧ suggestedTest(?p, Blood_test) ∧ suggestedTest(?p, Echocardiography_tests) ∧ suggestedTest(?p, Chest_X-ray) ∧ suggestedTest(?p, Electrocardiography_tests)
Patient Diagnosis Rules (PDR) - "Diagnosis"	Systolic heart failure diagnosis	PDR1: Patient(?p) ∧ hypotheticalDiagnosis(?p, Chronic_heart_failure) ∧ hasTestResults(?p, Chest_X-ray_abnormal) ∧ hasTestResults(?p, ECG_abnormal) ∧ hasTestResults(?p, BNP_value_higher_than_100_pg_per_ml) ∧ hasTestResults(?p, LVEF_lower_than_40_percent) → suggestedDiagnosis(?p, Systolic_heart_failure)
	Hypertensive heart failure diagnosis	PDR2: Patient(?p) ∧ hypotheticalDiagnosis(?p, Chronic_heart_failure) ∧ hasSigns (?p, High_blood_pressure_disorder) ∧ hasSigns (?p, Abnormal_heart_sounds) ∧ hasTestResults(?p, ECG_abnormal) ∧ hasTestResults(?p, Pulmonary_venous_congestion) → suggestedDiagnosis(?p, Hypertensive_heart_failure)
Patient Treatment Rules (PTR) -"Treatment"	Systolic heart failure treatment	PTR1: Patient(?p) ∧ suggestedDiagnosis(?p, Systolic_heart_failure) → suggestedTreatment(?p, ACE_inhibitor) ∧ suggestedTreatment(?p, Beta_blocker)
		PTR2: Patient(?p) ∧ suggestedDiagnosis(?p, Systolic_heart_failure) ∧hasSymptoms(?p, edema) ∧ hasSymptoms(?p, congestion) → suggestedTreatment(?p, ACE_inhibitor) ∧ suggestedTreatment(?p, Beta_blocker) ∧ suggestedTreatment(?p, Diuretic)
Prescription Checking Rules (PCR) -"Prescription"	Patient-drug interaction	PCR1: Patient (?p) ∧ allergicTo (?p, Aspirin) ∧ Prescription (?Rx) ∧ hasPrescription (?p, ?Rx) swrlb:contains (?Rx, Aspirin) → hasAlert (?p, This patient is allergic to Aspirin)
	Drug-drug interaction	PCR2: Patient (?p) ∧ hasPrescription (?p, ?Rx1) ∧ Prescription (?Rx1) ∧ Prescription (?Rx2) ∧ interactWith (?Rx1, ?Rx2) → forbidRx (?p, ?Rx2)

Figure 9 shows the translation of ontologies and SWRL rules to enable this process. Here, SWRL rules are stored as OWL individuals with their associated knowledge base. First, SWRL rules are translated into Jess rules using a SWRLJessBridge [36], which is a Protégé plug-in, and added to the Jess

rule engine; then, the ontologies and the knowledge base are translated into Jess facts and introduced into the rule engine as well; third, Jess rule engine enables the reasoning and produces results in the Jess format; and finally these results are translated back into the OWL format. In this figure, we have shown the format of Jess facts as well as the format of Jess rules for PTR1 from Table 2.

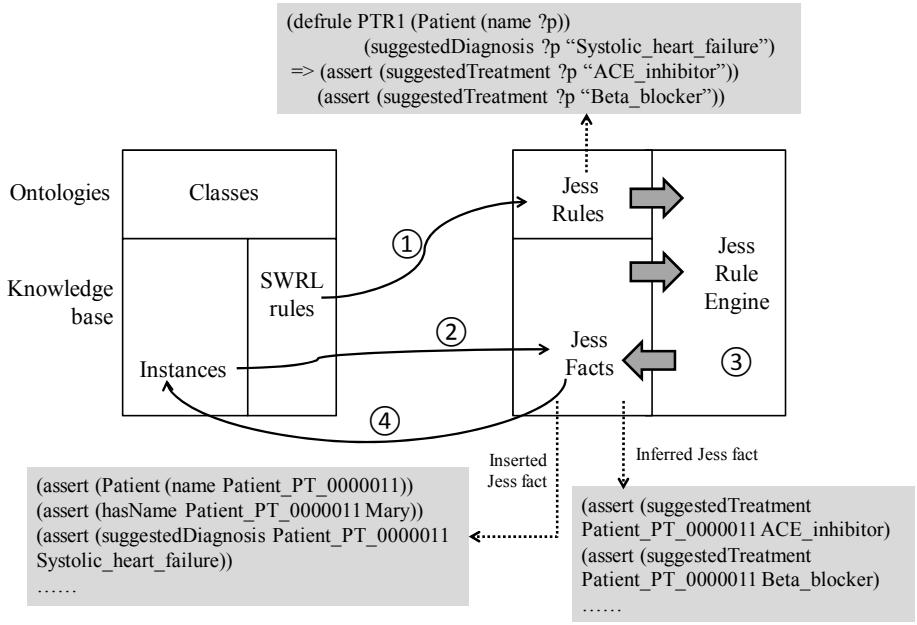
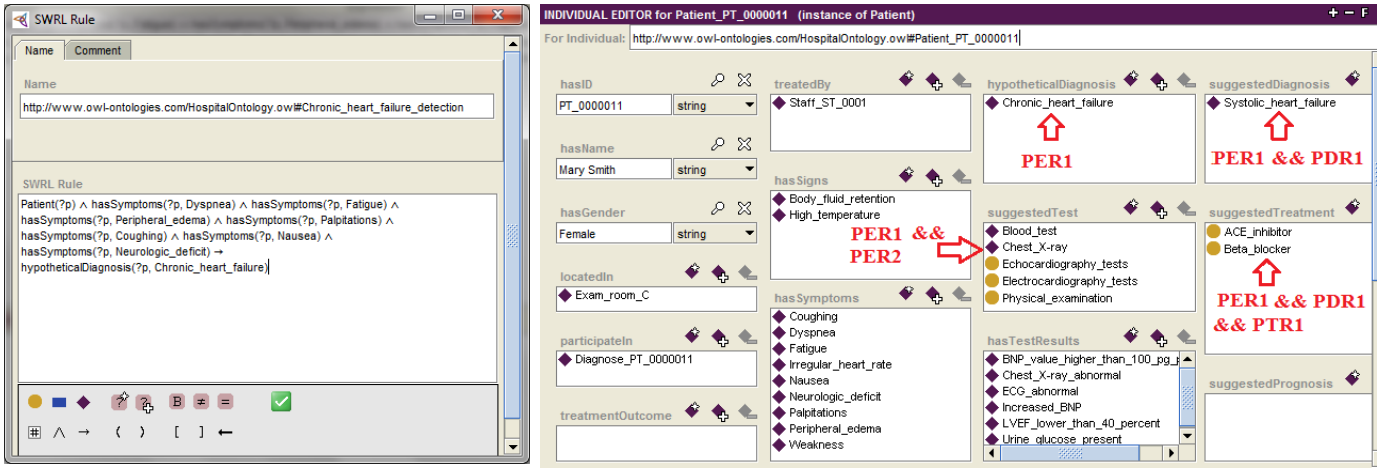


Figure 9. SWRL reasoning using Jess rule engine

SWRL rules can be easily modeled in SWRLJessTab [36] which is a Protégé plug-in and provides a graphical interface to interact with SWRLJessBridge. SWRLJessTab allows us to insert, remove and edit SWRL rules. Figure 10 (a) shows the implementation of rule PER1 in this editor. With the symptoms asserted by a nurse (see Figure 10 (b)), this rule infers that the patient may have chronic heart failure, as shown in Figure 10 (b). Meanwhile, another triggered PER2 rule uses results generated by PER1 to induce forward chaining and produces suggested tests as output. These results are presented to a doctor during physical examination. As this clinical process proceeds other rules may be triggered as more data becomes available and is asserted into the knowledge base. For example, signs are asserted by the doctor and lab test results are asserted by lab staff. Accordingly, treatments are recommended, such as ACE inhibitor and Beta blocker medication.

In this example, rule-based reasoning is conducted recursively, which means a rule can use the results of another rule as input data until no rules can be fired. This process is called *forward chaining* and is widely used in rule-based systems. Thus, the knowledge base is kept continuously updated by the inserted facts. The results of rule-based reasoning are tightly integrated with clinical pathways and provide evidence-based decision support for clinicians through a user-friendly interface. In the next section, we discuss how the knowledge derived from SWRL rules is used to implement flexible clinical pathways.



(a) Rule in SWRL editor (b) Results of SWRL rule chaining

Figure 10. Encoding and reasoning of chronic heart failure using SWRL rules

5. INTEGRATION OF CLINICAL PATHWAY AND RULES FOR FLEXIBILITY

The key innovation in CONFlexFlow is the tight integration of pathways and rules to improve the operation of a CDSS, while recognizing that at some stages in a clinical workflow the subsequent path is determined by the context. Hence, flexibility is of the essence. Actually, the decision on the next path to take is influenced by both the medical plan and the contextual information of an individual patient. The context is derived not only from the current case data, but is also based on the specific pathway that was taken to reach this point in the workflow. Here, we first give a framework for integration and then discuss our implementation in detail.

5.1. A Framework for Integration

There is a *m-to-n* relationship between (activity, context) on the one hand, and rules on the other. This means a certain contextual situation with regards to an activity may need the application of many rules, while a given rule may apply to many (activity, context) situations. Thus, a context where a patient is not allergic to any substance will not trigger allergy check rules, and similarly if a patient does not carry insurance, an insurance check is omitted. Some examples of these "(Activity, Context) → Rules" are:

```
(Medication, allergic) → allergy check rule(s)
(Admission, broken leg) → wheel chair check rule(s)
(Admission, emergency)→ emergency procedure rule(s)
(Do tests, X-ray) → dose check
(Revaluation, overweight)→ diet recommendation rule(s)
```

Hence, if a patient exhibits allergies to some drug during the medication activity, then an allergy check is needed. At the admission time, if there is an emergency then a different admission procedure from the normal one is followed. But clearly, at admission time we do not need to check whether the patient has an allergy. Thus, only a small set of rules related to a specific activity should be triggered before the activity is executed according to the current context.

Figure 11 presents our proposed methodology for integrating context and rules to design a customized subprocess at runtime. The main idea is that after relevant rules are extracted based on the (context, activity) combination as discussed above, they are fired to determine the execution semantics for the isolated activities contained in the loosely coupled composite activity. Thus, a subprocess fragment is generated and executed as the subsequent pathway for the currently running process instance.

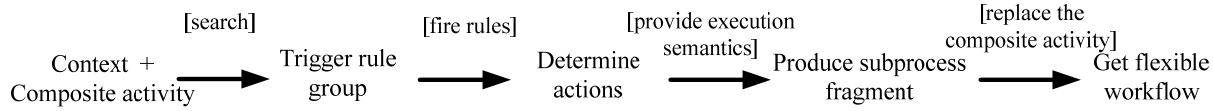


Figure 11. Methodology for integration of context, activity, and rules

5.2. Designing a Flexible Clinical Workflow using BPMN 2.0

BPMN 2.0 [38] not only defines a standard for how to graphically represent a process model, but also includes execution semantics for the defined elements. It uses an XML format to store and share process

definitions. This new standard provides a large variety of node types including events, activities, gateways, etc. Specifically, we use *human task*, *rule task*, *reusable subprocess* and *ad hoc subprocess* frequently in our implementation. Their semantics have been introduced earlier. A *human task* is very useful for modeling clinical scenarios since most medical activities cannot be completely automated, but instead require a lot of user interaction. The recommendations and reminders are presented to clinicians through a user friendly interface. A *reusable subprocess* refers to those strictly defined processes that can be reused in many scenarios, such as ECG and X-ray process. Moreover, we use an *ad hoc subprocess* to realize our notion of workflow flexibility because it allows context-aware customization of a loosely defined composite activity.

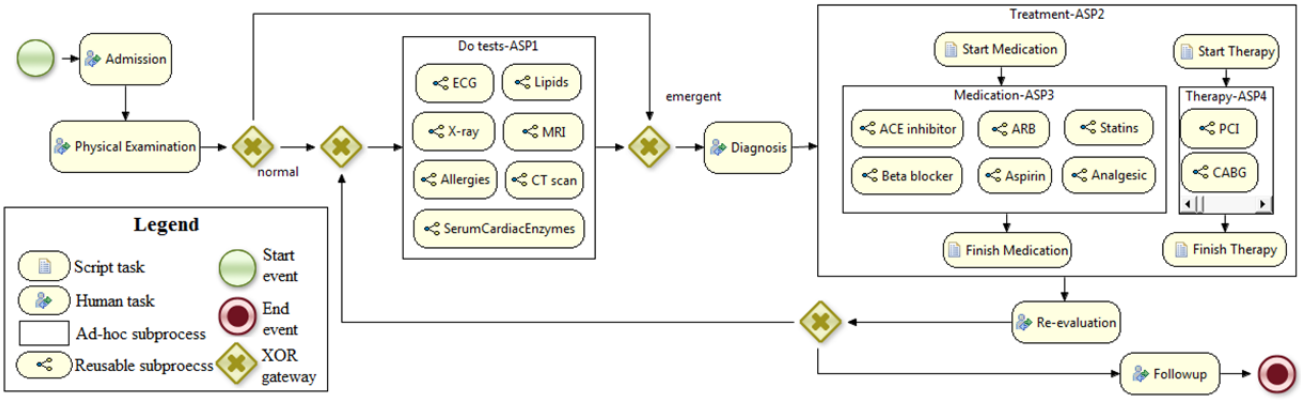


Figure 12. A loosely coupled BPMN workflow implemented in Drools-flow [5]

Figure 12 shows such a BPMN 2.0 workflow model for treating heart diseases and it is implemented using Drools-flow [5]. We make medical plans or protocols, which are originally presented in flow charts or plain text, executable through workflow modeling. An example of such guidelines for treating heart attack patients is presented in Appendix A. This process has several human tasks, e.g., "Physical examination" and "Diagnosis," which require interaction with clinicians. Clinical data is obtained from human tasks by user input and is then accessible to the workflow engine. In particular, we have four ad hoc subprocesses (ASP) for composite activities to be customized at runtime. "Do tests–ASP1," "Medication–ASP3," and "Therapy–ASP4" are three composite activities that contain a number of atomic activities, while "Treatment–ASP2" is a *nested ASP* with other ASP's nested inside it. Whether the

contained activities will be executed, and if so in what order they are executed, is unknown at design time. For example, "Do tests–ASP1" refers to the set of scheduling and testing activities that are likely to follow the doctor's examination. *But, it is not possible to predict what tests the doctor will prescribe until the patient is examined.* Similarly, the therapy activity is loosely defined and it may include Percutaneous coronary intervention (PCI) and Coronary artery bypass grafting (CABG) depending upon the diagnosis. Hence, the execution semantics for instantiating an ASP is only known when this node is reached. A concretely defined subprocess will be produced dynamically based on the new data inserted into the context base. Similarly, the procedure for patient treatment depends on her test results, further diagnosis and medical history.

5.3. Workflow Engine – Drools-Flow 5.2

We used an open source tool, Drools-Flow 5.2 (also known as jBPM5) [5], as the workflow engine. It implements most element types defined in BPMN 2.0, and allows us to execute the clinical process in Figure 12. In addition, we used Drools-Expert [5], which is essentially a rule engine, as a complementary tool to support the rerouting of subsequent pathways. Figure 13 shows the CONFlexFlow implementation using the Drools framework.

We created a stateful knowledge session that will load required BPMN process models and the Drools rule files into the production memory at runtime. In this study, we have the heart failure process (i.e., HeartFailureProcess.BPMN), ECG process (i.e., ECGProcess.BPMN), and some other medical processes in BPMN files. Besides, heart failure rerouting rules (i.e., HeartFailureReroutingRules.drl) are loaded to demonstrate workflow flexibility. Drools rule files have a *.drl* (Drools Rules Language) extension. The Drools workflow engine manages the status of process instances along with their associated data. The Drools rule engine is informed about processes by inserting their current state into the working memory. On the other hand, the actions determined by the Jess Rule Engine for the current process instance are passed to the workflow engine through the working memory.

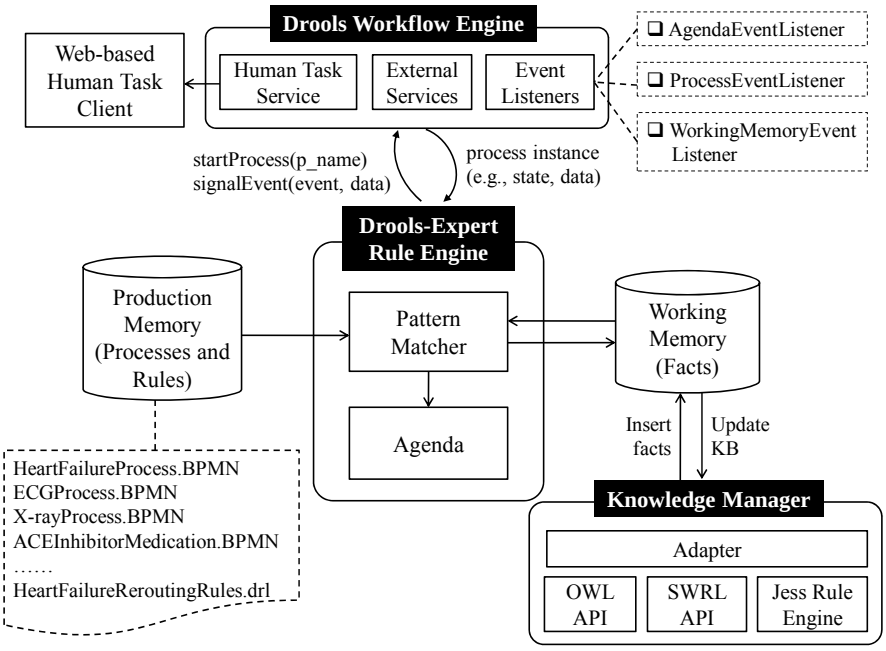


Figure 13. CONFlexFlow implementation using the Drools framework [5]

In the Drools framework, Drools-Expert is a forward chaining inference engine that implements an enhanced and extended RETE algorithm [20], called ReteOO. It will then be able to derive the next steps taking Drools rules and processes into account jointly. If a part of the process needs to be executed, the rule engine will request the workflow engine to execute that step. This can be done easily since the process engine is equipped with a *WorkingMemoryEventListener (WMEL)* that "listens" for any events sent from the rule engine. Once the current step is completed, the process engine returns control to Drools-Expert to again derive the next step. The Drools implementation gives the control to the rule engine to decide and notify the workflow engine of what step(s) to do next. Thus, the Drools rules allow us to alter process behavior dynamically.

The Drools workflow engine comes with several event listeners for responding to different types of events. For example, upon triggering a signalEvent the WMEL listener can activate any event or activity in the running process instance. Event related data can be passed from the parent process to a subprocess or activity as well. The *ProcessEventListener* "listens" for events from current instances, such as *beforeNodeTriggered*, *beforeProcessStarted* and *afterProcessStarted*. They provide real time updates on

process status and are widely used to support medical tasks. For example, after the human task node "Physical Examination" is triggered, the system gets results from medical reasoning rules and prepares the data to be shown to clinicians. The *AgendaEventListener* is aware of rule creation, activation, firing, etc. The Drools rules can be updated (i.e., inserted or deleted) in the production memory at runtime, and such events should be captured by the system to provide a response.

The *Agenda* manages the execution order of conflicting rules using the conflict resolution strategy. The default strategies employed by Drools are *salience* and *LIFO* (Last in, First out). We use salience to specify the priority of rules, and the one with the higher salience value is preferred. In addition, we use a *ruleflow-group* to associate a set of Drools rules to a rule task or an ad hoc subprocess. Any rule has the attribute *ruleflow-group* with the same name as the activity that will be triggered before the activity node is activated. The workflow process will immediately continue with the next node if it encounters a *ruleflow-group* where there are no active rules at that point. To allow clinicians to interact with clinical process instances, we use the web-based process management console provided by Drools. They can input clinical data that will be used in the subsequent pathways.

5.4. Customizing an Ad hoc Subprocess at Runtime

Next, we show how to realize the execution semantics for an ad hoc subprocess at runtime. Figure 14 shows the representation of the ad hoc subprocess "Treatment-ASP2" in BPMN 2.0 XML. It is nested with two other ASPs "Medication-ASP3" and "Therapy-ASP4", which must be carried out in sequence, while the activities within these two ad hoc subprocesses occur in parallel. This is specified using the attribute *ordering*. A subprocess is completed when there is no active instance running (i.e., specified by the attribute *completionCondition*). We can see that for an ad hoc subprocess, there is no connection between containing nodes (i.e., activities) thus their execution order is not known at design time. There can be a number of execution semantics for connecting these nodes and they are specified in a rule group with the same name (i.e., "Treatment-ASP2"). The activation of rule groups is maintained by the *Agenda* and the workflow engine is notified of rule activation by the *AgendaEventListener* (see Figure 13).

```
<!--ad hoc subprocess for treatment activity, contained activities can execute in sequential-->
<adHocSubProcess id="_8" name="Treatment-ASP2"  ordering="Sequential" >
    <!-- nodes -->
<!--ad hoc subprocess for therapy activity, contained activities can execute in parallel-->
<adHocSubProcess id="_8-2" name="Therapy-ASP4"  ordering="Parallel" >
    <!-- nodes -->
    <callActivity id="_8-2-1" name="Coronary therapy" calledElement="CornoaryTherapy"/>
    <callActivity id="_8-2-2" name="Bypass surgery" calledElement="BypassSurgery" />
    <!-- connections -->
<!—There can be no connection within an ad hoc sub-process-->
    <completionConditionxsi:type="tFormalExpression">
    getActivityInstanceAttribute("numberOfActiveInstances") == 0
    </completionCondition>
</adHocSubProcess>
<!--ad hoc subprocess for medication activity, contained activities can execute in parallel-->
<adHocSubProcess id="_8-1" name="Medication-ASP3"  ordering="Parallel" >
    <!-- nodes -->
    <callActivity id="_8-1-1" name="ACE inhibitor" calledElement="ACEInhibitor" />
    <callActivity id="_8-1-2" name="ARB" calledElement="ARB" />
    <callActivity id="_8-1-3" name="Beta blocker" calledElement="BetaBlocker" />
    <callActivity id="_8-1-4" name="Aspirin" calledElement="AspirinMedication" />
    <callActivity id="_8-1-5" name="Statins" calledElement="StatinsMedication" />
    <callActivity id="_8-1-6" name="Analgesic" calledElement="AnalgesicMedication"/>
    <!-- connections -->
    <completionConditionxsi:type="tFormalExpression">
    getActivityInstanceAttribute("numberOfActiveInstances") == 0
    </completionCondition>
    </adHocSubProcess>
<scriptTask id="_8-3" name="Start Medication" >
<script>System.out.println("Start medication");</script>
</scriptTask>
    ......
    <!-- connections -->
    <sequenceFlow id="_8-3-_8-1" sourceRef="_8-3" targetRef="_8-1" />
    <sequenceFlow id="_8-5-_8-2" sourceRef="_8-5" targetRef="_8-2" />
    <completionConditionxsi:type="tFormalExpression">
    getActivityInstanceAttribute("numberOfActiveInstances") == 0
    </completionCondition>
</adHocSubProcess>
```

Figure 14. XML representation of the nested ad hoc subprocess "Treatment-ASP2" in Figure 12

Figure 15 shows sample rules associated with their ad hoc subprocess (comment lines start with //). The first rule R1 is associated with the "Treatment-ASP2" subprocess, since they belong to *ruleflow-group* "Treatment-ASP2". This rule shows that therapy and medication will only be carried out when a patient needs thrombus (or blood clot) breaking. Similarly, R2 and R3 belong to the "Medication-ASP3" activity. Rule "R2-General Medication" will always be triggered but it will assign different medications (i.e., ACE inhibitor or ARB) depending on whether the patient is intolerant of ACE inhibitor. R3 will prescribe statins (or HMG-CoA reductase inhibitors) when a patients has high Low-Density Lipoprotein (LDL). R2 is given a higher priority than R3 by using the *salience* attribute since ACE inhibitor or ARB medication should be given first to heart failure patients to control their blood pressure, treat heart failure and prevent strokes. Our implementation has more such rules for handling different kinds of scenarios.

```
rule"R1-Treatment for patients require thrombus breaking"ruleflow-group"Treatment-ASP2"
when
processInstance: WorkflowProcessInstance()
then
    if(processInstance.getVariable("requireThrombusBreaking").equals("yes"))
      //triggering both therapy and medication (in sequence) if thrombus breaking is required
    drools.getContext(ProcessContext.class).getProcessInstance().signalEvent("Start Therapy", pdata);
    drools.getContext(ProcessContext.class).getProcessInstance().signalEvent("Start Medication", pdata);
    else
    //triggering only medication activity if thrombus breaking is not required for this patient
    drools.getContext(ProcessContext.class).getProcessInstance().signalEvent("Start Medication", pdata);
end

rule"R2-General medications"ruleflow-group"Medication-ASP3"
salience30
when
processInstance: WorkflowProcessInstance()
then
    if (processInstance.getVariable("intolerantOfACE_inhibitor").equals("no"))
       drools.getContext(ProcessContext.class).getProcessInstance().signalEvent("ACE inhibitor", data);
    else
      //ARB medication is usually given to patients if they are intolerant of ACE inhibitor
      drools.getContext(ProcessContext.class).getProcessInstance().signalEvent("ARB", pdata);
end

rule"R3-Medication for patient with high LDL"ruleflow-group"Medication-ASP3"
salience 20
when
processInstance: WorkflowProcessInstance()
eval(processInstance.getVariable("hasHighLDL").equals("yes"))
then
    drools.getContext(ProcessContext.class).getProcessInstance().signalEvent("Statins");
end
```

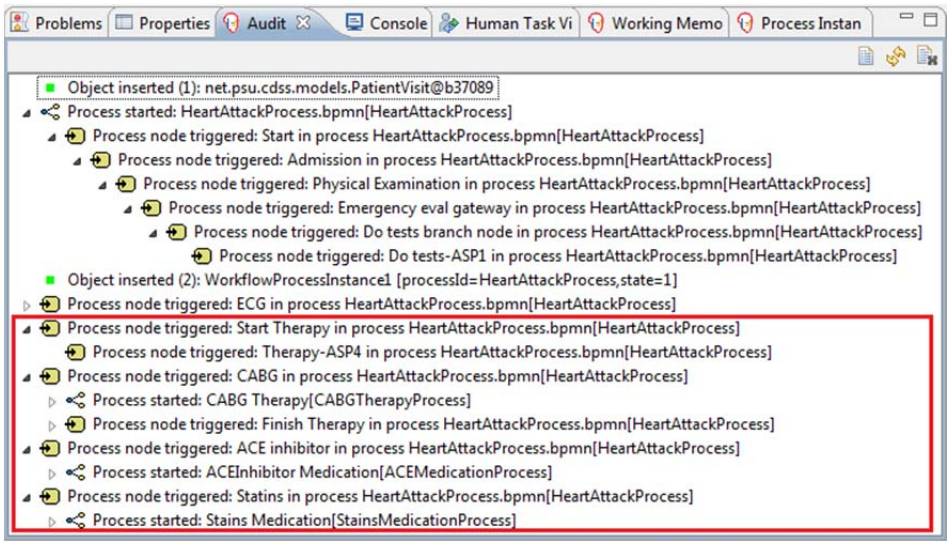
Figure 15. Rules associated with the "Treatment"and "Medication" ad hoc subprocess

In this way, a variety of scenarios can be created for an ad hoc subprocess according to different contexts. Figure 16 (a) shows the process log generated by Drools when the workflow model in Figure 12 is instantiated for a specific patient who needs thrombus breaking, is not intolerant of ACE inhibitor and shows signs of high LDL, etc. It reflects the operational semantics for the ASP "Do-tests-ASP1" and "Treatment-ASP2" in a particular context. The visualization of the actual pathway for treating this patient (i.e., "Treatment-ASP2") is presented in Figure 16 (b). More complicated workflow patterns can be modeled and instantiated in this approach as well.

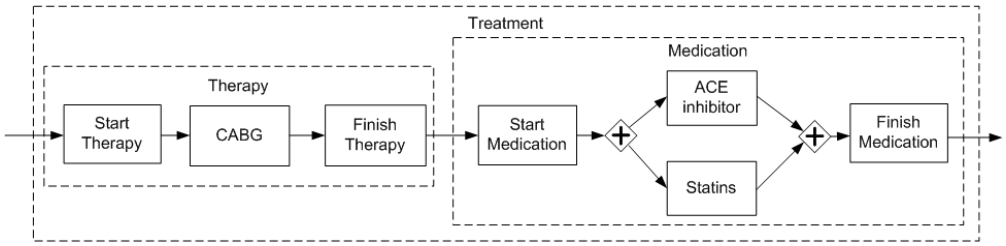
6. DISCUSSION

Above we have described a novel and practical framework CONFlexFlow for designing a clinical context and guideline integrated CDSS. We argue that flexible integration of CDSS with a clinical workflow is a key to its success. Moreover, semantic web technologies like OWL can help to create ontologies that are exchangeable across various healthcare departments and organizations. This promotes the understanding

of medical knowledge across different providers and also enables sharing and extensibility. Thus, one provider can import an existing ontology and extend it further without having to reinvent the wheel. This approach is formal, yet also flexible.



(a) Drools screenshot of the process log for a specific patient



(b) Visualization of the runtime results for ad hoc subprocess "Treatment-ASP2" in Figure 12

Figure 16. Results of the ad hoc subprocess instantiation

6.1. Results of Ontology Analysis

The clinical context model is developed following a formal methodology [35]. First, we enumerated healthcare use cases and checks on existing ontologies. Then, we defined the five top-level classes and 43 sub-classes in the hierarchy. The third step involved creating all the properties for the existing classes, including 47 Object properties and 106 Datatype properties. Their domain and range were then defined accordingly. Next, 126 restrictions were created for all the classes to enrich the ontology semantically. When the ontology schema was completed, we added 30 patients, 25 hospital personnel, 40 assets, and 40 locations as test cases. Our implementation is available on the web [54] for further extension.

To validate our model, we installed Pellet Reasoner Plug-in [42] for Protégé 3.4. This tool can check an OWL-encoded ontology for inconsistencies and infer new instances or classes. After each iteration of ontology development, we ran Pellet against the ontology to check its validity and revised it if any inconsistency or redundancy existed. The final result showed that our context ontology was valid in terms of logical consistency, concept satisfaction and classification. In our knowledge base, we also include the heart failure ontology developed by the HEARTFAID team [4]. This ontology contains more than 200 classes, 100 properties, and 1000 individuals. Note that for now we only focus on the diagnosis and treatment of heart failure patients based on context information. Of course, other medical domain ontologies can also be integrated into our model in a similar way. Reusing mature existing medical ontologies can make the model more complete for real environments.

Based on these ontologies, we developed 18 SWRL rules for describing heart failure procedural knowledge. They include the detection, diagnosis and treatment of chronic heart failure, systolic heart failure, hypertensive heart failure, etc. The Jess Rule engine is used to automate the reasoning process. We aim to add more semantic rules in future to cover more complete medical knowledge.

6.2. Contributions, Success factors and KPIs

Another contribution of this work is the tight integration of flexible clinical pathway with medical decision support. Our study differs from contemporary workflow modeling approaches in the following ways: (1) we use a standard process modeling language BPMN 2.0 to model the care process. It provides rich semantics (e.g., human task, rule task and subprocess) for modeling clinical activities and coordinates interactions among various healthcare entities; (2) Clinical processes are strictly defined in the overall structure, yet some knowledge intensive activities are loosely defined using ad hoc subprocesses. The actual execution semantics are triggered by the clinical context that is derived from previous medical tasks and the current environment. This contextual information is only available at runtime and is unique for each individual patient. In addition, an ad hoc subprocess can be nested to handle complicated, dynamic scenarios; and (3) the decision support provided during patient encounters is based on formal

context and medical ontologies that are aligned with clinical guidelines. Thus, we can ensure the correctness of medical recommendations.

Many studies have discussed the factors leading to successful CDSS implementation [11, 43, 49]. The most critical factors are: (1) capture of evidence in machine-interpretable knowledge base; (2) computerized decision support instead of paper-based; (3) timely advice; (4) workflow integration; (5) response to user needs; (6) maintenance and extension; and (7) clinical effects and costs. Our ConFlexFlow framework has covered most of these aspects but the evaluation of our system in terms of user satisfaction is out of the scope of this paper. A key objective in developing and implementing such a system is to improve quality as reflected in concrete measures. Although detailed discussion of quality is beyond the scope of the current work, some key measures of quality (KPI's) for our purposes are: *number of treatment errors because of drug interactions (or allergies), number of diagnosis errors, number of cases of treatment not covered by patient's insurance, number of treatment failures for lack of available resources, complication rate per patient, patient satisfaction, etc.* These metrics will inform the evaluation of the impact of the proposed system. One way to use these metrics is by means of a post hoc audit. For example, we can compare the number of treatment errors from drug interactions after the CDSS is in place versus the corresponding number before installing such a system by examining the patient log over two different periods of equal lengths.

7. CONCLUSIONS

The goal of this research is to study new approaches for designing clinical decision support systems. We proposed a framework called ConFlexFlow, and showed how flexible and adaptable clinical pathways can be designed taking into account medical knowledge in the form of rules and detailed contextual information to achieve a high quality outcome. A clinical workflow charts a path for a patient through the various steps in interacting with a PCP office, lab, pharmacy and other participants in the process of patient care. These pathways are selected during workflow execution based on rules that encapsulate medical knowledge and patient conditions that are constantly changing. The ontologies described in this

paper were implemented in Protégé 3.4. Furthermore, we developed a proof of concept prototype using the Drools framework.

Although we use SWRL rules to model medical knowledge, we further plan to use a mapping mechanism to make the knowledge convertible between an Arden-like syntax and SWRL to make our approach more acceptable in the medical domain. Next, we plan to enhance the current prototype and test it in a practical environment using the proposed quality metrics. An audit of past errors without a CDSS, and errors under the CDSS operation, can reveal the improvement realized from the new system. Other issues of interest are, how to create a contextual summary automatically for a doctor based on a patient's record, and how to allow a doctor to customize her alert settings so the alerts are useful but not distracting.

Thus, the proposed research poses several technical challenges and promises many benefits to the healthcare community at a time when costs are running out of control. It can lead to better, "smarter" routing of clinical workflows by applying rules with a deeper understanding of context. Moreover, various kinds of alerts can help to reduce the incidence of treatment errors and lead to improved patient safety. Finally, better and quicker recommendations can be generated at various decision points.

Acknowledgment

This work was supported in part by the Center for Integrated Healthcare Delivery Systems (CIHDS) and by the Smeal College of Business at Penn State.

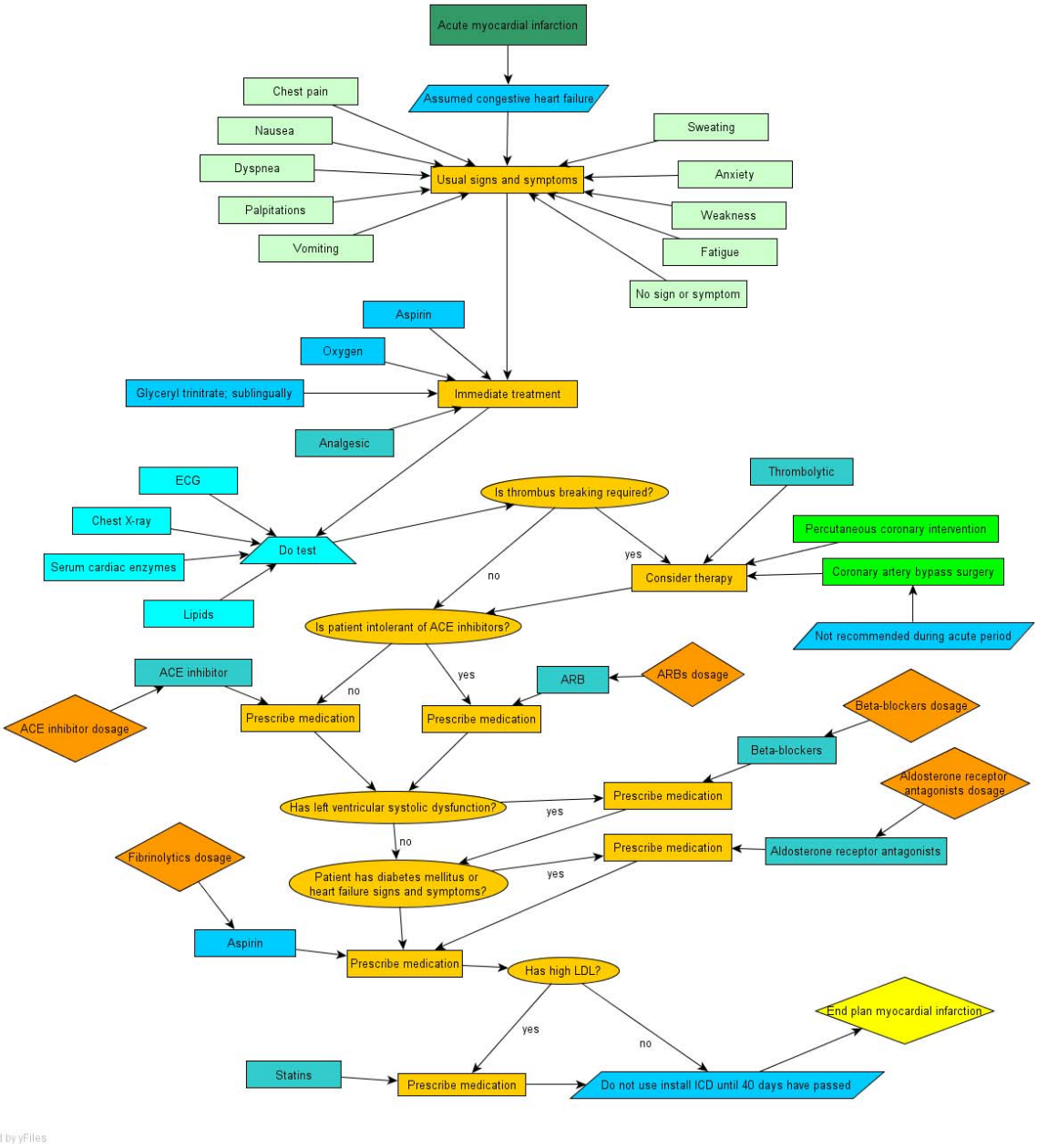
REFERENCES

[1] American Academy of Family Physicians (AAFP) in, (http://www.aafp.org/).
[2] American Heart Association., in, (http://www.heart.org/).
[3] Signavio, in, (http://academic.signavio.com).
[4] HF Ontology, in, (University of Calabria, Department of Electronics, informatics, Systems (DEIS), Italy, 2008).
[5] Drools - The Business Logic integration Platform. http://www.jboss.org/drools, in, (Jboss Community, 2011).
[6] S.R. Abidi, et al., Ontology-based modeling of clinical practice guidelines: a clinical decision support system for breast cancer follow-up interventions at primary care settings, Studies in Health Technology and Informatics, 129(Pt 2) (2007) 845-849.
[7] M. Adams, et al., Extensible and context-aware exception handling for workflows, In: Proceedings of CoopIS'07, (2007), pp. 95-112.

[8] D. Alexandrou, et al., SEMPATH: semantic adaptive and personalized clinical pathways, in: 2009 International Conference on eHealth, Telemedicine, and Social Medicine, (2009), pp. 36-41.
[9] L. Ardissono, et al., Context-aware workflow management, in: P.F. L. Baresi, G.-J. Houben (Ed.) ICWE, (2007), pp. 47-52.
[10] S. Bechhofer, et al., OWL web ontology language reference, W3C recommendation, 10(2004) 2006–2001.
[11] E.S. Berner, Clinical decision support systems: State of the Art, in: AHRQ Publication No. 09-0069-EF, (Rockville, Maryland: Agency for Healthcare Research and Quality, 2009).
[12] O. Bodenreider, The unified medical language system (UMLS): integrating biomedical terminology, Nucleic acids research, 32(suppl 1) (2004) D267–D270.
[13] M. Ceccarellia, et al., A Guideline Engine For Knowledge Management in Clinical Decision Support Systems (CDSSs), in: Proceedings of SEKE, (2009), pp. 252-257.
[14] B. Chen, et al., Analyzing medical processes, in: Proceedings of the 30th international conference on Software engineering, (Leipzig, Germany, 2008), pp. 623-632.
[15] L.A. Clarke, et al., Using software engineering technology to improve the quality of medical processes, in: Companion of the 30th international conference on Software engineering, (Leipzig, Germany, 2008), pp. 889-898.
[16] J. Dang, et al., An ontological knowledge framework for adaptive medical workflow, Journal of Biomedical Informatics, 41(5) (2008) 829-836.
[17] J. Dang, et al., Personalized medical workflow through semantic Business Process Management, in: 11th International Conference on Enterprise Information Systems (ICEIS), (2009).
[18] A.K. Dey, Understanding and using context, Personal Ubiquitous Comput., 5(1) (2001) 4-7.
[19] M. Fieschi, et al., Medical decision support systems: old dilemmas and new paradigms, Methods Inf Med, 42(3) (2003) 190–198.
[20] C. Forgy, Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence, 19(1982) 17-37.
[21] E. Friedman-Hill, others, Jess, the rule engine for the Java platform, Sandia National Laboratories, (2005).
[22] M. Heravizadeh, D. Edmond, Making workflows context-aware: a way to support knowledge-intensive tasks, in: Proceedings of the fifth on Asia-Pacific conference on conceptual modelling, (Wollongong, NSW, Australia, 2008), pp. 79-88.
[23] HL7, Health Level Seven. http://www.hl7.org/
[24] I. Horrocks, et al., SWRL: A semantic web rule language combining OWL and RuleML, W3C Member submission, 21(2004).
[25] D. Isern, A. Moreno, Computer-based execution of clinical guidelines: A review, International Journal of Medical Informatics, 77(12) (2008) 787-808.
[26] D.O. Kang, et al., A wearable context aware system for ubiquitous healthcare, in: 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS'06), (2006), pp. 5192–5195.
[27] B.T. Karsh, Clinical practice improvement and redesign: how change in workflow can be supported by clinical decision support, in: AHRQ Publication No. 09-0054-EF, (Agency for Healthcare Research and Quality, Rockville, Maryland, 2009).
[28] K. Kawamoto, et al., Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success, British Medical Journal, 330(7494) (2005).
[29] E.J. Ko, et al., Ontology-based context modeling and reasoning for U-HealthCare, IEICE TRANSACTIONS on Information and Systems, 90(8) (2007) 1262–1270.
[30] L.T. Kohn, et al., To err is human: building a safer health system, National Academy Press, Washington DC, (2000).
[31] R. Lenz, M. Reichert, IT support for healthcare processes-premises, challenges, perspectives, Data & Knowledge Engineering, 61(1) (2007) 39–58.

[32] J. Mathe, et al., Model-based design of clinical information systems, Methods of Information in Medicine, 47(5) (2008) 399-408.
[33] J.L. Mathe, et al., A Model-Integrated, Guideline-Driven, Clinical Decision-Support System, in: IEEE Software, (2009), pp. 54-61.
[34] R. Müller, et al., Agentwork: a workflow system supporting rule-based workflow adaptation, Data & Knowledge Engineering, 51(2) (2004) 223–256.
[35] N.F. Noy, D.L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology, (2001).
[36] M.J. O'Connor, et al., Supporting Rule System Interoperability on the Semantic Web using SWRL and Jess, in: Fourth International Semantic Web Conference (ISWC2005), (Galway, Ireland, 2005), pp. 974-986.
[37] L.a.G. Ohno-Machado, J.H. and Murphy, S.N. and Jain, N.L. and Tu, S.W. and Oliver, D.E. and Pattison-Gordon, E. and Greenes, R.A. and Shortliffe, E.H., Barnett, G., The Guideline Interchange Format (GLIF), Journal of the American Medical Informatics Association, 5(4) (1998) 357.
[38] OMG, Business Process Model And Notation (BPMN) Version 2.0, in, (January 2011).
[39] OpenClinical, Arden Syntax. http://www.openclinical.org/gmm_ardensyntax.html.
[40] L.J. Osterweil, et al., Engineering Medical Processes to Improve Their Safety, in: IFIP International Federation for Information Processing, (Boston Springer, Geneva, Switzerland, 2007), pp. 267–282.
[41] F. Paganelli, D. Giuli, An ontology-based context model for home health monitoring and alerting in chronic patient care networks, In: 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), (Niagara Falls, Ontario, Canada, 2007), pp. 838-845.
[42] B. Parsia, E. Sirin, Pellet: An owl dl reasoner, In: Third International Semantic Web Conference-Poster, (2004).
[43] M. Peleg, S. Tu, Decision support, knowledge representation and management in medicine, Yearbook of medical informatics, (2006) 72-80.
[44] M. Peleg, et al., Comparing models of data and knowledge for guideline-based decision support, in: Report SMI-2002-0923, (2002).
[45] M. Peleg, et al., Comparing computer-interpretable guideline models: A case-study approach, Journal of the American Medical Informatics Association, 10(1) (2003) 52-68.
[46] A.L. Rector, et al., The GALEN high level ontology, Studies in Health Technology and Informatics, (1996) 174–178.
[47] M. Reichert, P. Dadam, ADEPT flex—supporting dynamic changes of workflows without losing control, Journal of Intelligent Information Systems, 10(2) (1998) 93–129.
[48] G. Schreiber, et al., Knowledge Engineering and Management: The CommonKADS Methodology. , in: The MIT Press, (Cambridge, MA, 2000).
[49] D.F. Sittig, et al., Grand challenges in clinical decision support, Journal of Biomedical Informatics, 41(2) (2008) 387-392.
[50] Standford, Protégé: an open source ontology editor and knowledge-based framework. http://protege.stanford.edu/.
[51] J.M. Teich, et al., Clinical decision support in electronic prescribing: recommendations and an action plan, British Medical Journal, 12(4) (2005) 365.
[52] R.L. Wears, M. Berg, Computer technology and clinical work: still waiting for Godot, The Journal of the American Medical Association (JAMA), 293(10) (2005) 1261-1263.
[53] A. Wise, Little-JIL 1.0 Language Report. Technical report (UM-CS-1998-024), in, (Department of Computer Science, University of Massachusetts, Amherst, MA 1998).
[54] W. Yao, Hospital ontology, in: http://www.personal.psu.edu/wxy119/hospital_ontology.owl, (2009).
[55] Y. Ye, et al., An ontology-based hierarchical semantic modeling approach to clinical pathway workflows, Computers in Biology and Medicine, 39(8) (2009) 722-732.
[56] L. Zhu, et al., Challenges Observed in the Definition of Reference Business Processes, in: BPM 2007 Workshops, (2007), pp. 95-107.

APPENDIX A
Medical plan for heart attack in flow chart
(Source: http://lis.irb.hr/heartfaid/plans)



Refer to http://lis.irb.hr/heartfaid/plans/PlanLegend.png for the legend