NOVA Workflow: A Workflow Management Tool Targeting Health Services Delivery

Wendy MacCaull and Fazle Rabbi

Centre for Logic and Information,
St. Francis Xavier University
{wmaccaul,rfazle}@stfx.ca

Abstract. We present the NOVA Workflow tool-suite, a prototype for a process, information and communication management tool to guide and inform real world workflows with special attention to the needs of health services delivery. NOVA Workflow is an innovative workflow management system which integrates formal verification into the software development process. For workflow modeling the tool uses the time Compensable Workflow Modeling Language (CWML_T) which produces reliable and structured workflow models and enhances error handling. The graphical editor of the tool gives a common platform for modeling, verifying and developing software. The SOA based architecture of the workflow engine ensures compliance with industry standards. The tool includes an automated translator to a model checking tool, a monitor to facilitate run-time compliance of (health care) policy, and a user friendly browser to give clinicians a convenient way to view a patient's information without losing the context. We propose an application of the browser to process diagnosis.

1 Introduction

This paper presents an integrated approach for modeling, verifying, developing and monitoring workflow management systems (WfMSs), with special attention to the needs of safety critical systems such as health care systems. A report estimated that approximately 98,000 deaths per year in the United States were the result of medical errors, many of which could be traced to faulty processes [18]. Errors not leading to death are costly and adversely affect the patient. WfMSs can help ensure compliance with protocols. Model checking processes in these systems, before enactment, can save time and reduce errors, while using a model checked monitor can alert clinicians to abnormal situations. However, commercial WfMSs do not have adequate rollback mechanisms (for error handling and side effects) and many model checkers deal only with relative (rather than quantified) time eg.: when an emergency case arrives at the hospital, standard model checking can only verify whether, for a particular process *"Eventually the patient receives the treatment"*, but to save the patient's life, it should be verified that *"The patient receives the treatment within half an hour"*.

Z. Liu and A. Wassyng (Eds.): FHIES 2011, LNCS 7151, pp. 75–92, 2012.

In this paper we present an integrated tool for developing and verifying enterprise software systems. Rather than verifying actual programs, an abstract specification for the software is written and used to verify properties of the system. The abstract specification is written using a limited syntax in Java and the specification is translated to a model for a model checker. Enterprise software usually consists of hundreds if not thousand of components; each component has many business logics, data access operations and third party service invocations. In addition, client applications require an enormous programming effort to provide sophisticated Graphical User Interfaces (GUI). To verify such complex software systems without abstraction is challenging. Our tool-suite NOVA Workflow[1] deals with this problem by abstraction (i.e., abstract process specification) and reduction which makes it feasible to verify enterprise and/or safety critical software systems.

The NOVA Workflow tool suite has five components, i) the NOVA Editor, ii) the NOVA Translator, iii) the NOVA Engine iv) the NOVA Monitor and v) the NOVA Browser. The NOVA Editor uses the graphical modeling language, the Compensable Workflow Modeling Language, extended with the time constraints of delay and duration (CWML_T). The NOVA Translator translates the workflow model and Java specification into DVE, the modeling language for the parallel distributed model checker DiVinE [5]. The NOVA Engine is a workflow engine based on Service Oriented Architecture (SOA). The NOVA Engine can be used in a system as a workflow library and it does not provide any restriction on application development. The engine was developed on the Spring [10] and Hibernate [9] platforms both of which can be deployed to various application servers. Spring is a widely used open source framework that helps developers build high quality applications faster. Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. We used a three tire architecture for NOVA Workflow as centralizing the business logic in an application tier has several advantages including maintainability, extensibility, and interoperability. The NOVA Monitor integrates time constrained monitors with workflow models. The NOVA Browser is a flexible user interface designed to allow brainstorming to enhance the user experience. The integrated tool support for modeling, verification and development of workflow management systems together with the monitor will greatly help its users build reliable safety critical systems. Fig. 1 shows the architecture of NOVA WorkFlow.

The rest of this paper is organized as follows. The components of the NOVA Workflow are described in section 2, (the NOVA Editor) section 3, (the NOVA Engine) section 4, (the NOVA Translator) section 5, (the NOVA Monitor) and section 6 (the NOVA Browser). Section 7 presents a case study and Section 8 discusses related work and concludes the paper. More details, case studies and proofs pertaining to the NOVA Editor, Translator, Engine and Monitor may be found in [30,27,29]. Most of the information on the NOVA Browser, including the proposed application, appears here in published form for the first time.

[1] `http://logic.stfx.ca/software/nova-workflow`

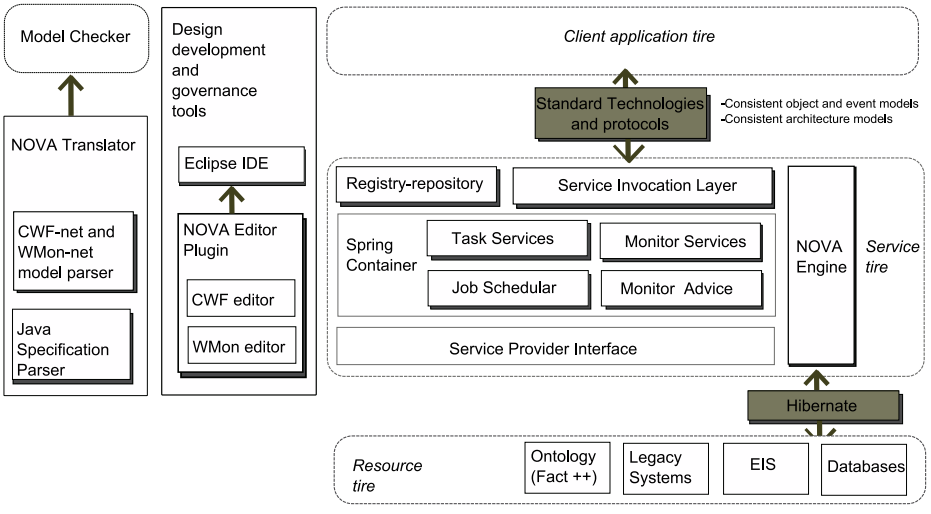

Fig. 1. SOA based architecture of NOVA workflow

2 The NOVA Editor

The NOVA Editor is a visual modeling tool for the time Compensable Workflow Modeling Language, CWML$_T$ [30,27]. A compensable workflow model consists of compensable and uncompensable tasks. In [30] we defined CWML (the untimed version of CWML$_T$). In addition to the basic operators "sequence" ($\bullet$), "and" ($\wedge$), "xor" ($\times$), "or" ($\vee$) and "loop" ($^+$), CWML uses the t-calculus operators [25] (sequential composition (;), parallel composition ($||$), internal choice ($\sqcap$), speculative choice ($\otimes$), and alternative choice ($\rightsquigarrow$)), to model compensation. In [27] we extended CWML with time, by including the notions of delay and duration and called it CWML$_T$. Timing constraints for most workflows can be expressed using delay and duration [26,19]. The foundations of CWML$_T$ are essentially time Petri nets (with integer valued time) referred to as Explicit Time Petri nets in [27]. Integers rather than reals suffice to model processes in health services delivery. A hybrid Petri net based semantics incorporating both weak and strong semantics is used [27] to model forward and compensation flows.

Atomic tasks in CWML$_T$ are of two types, uncompensable and compensable. An uncompensable atomic task is an activity which always finishes successfully, if activated. In case of an error executing the forward flow, a compensable task aborts and performs some compensation. The Petri net based representation of an atomic uncompensable task and an atomic compensable task with time constraints are given in Fig. 2. The Petri net representations of compound tasks may be found in [30,29].

In Fig. 2 solid arcs represent a forward flow and dotted arcs represent a compensation flow; d_1, d_2 and d_3 are the *delay*, *duration*, and *compensation duration* respectively. *Delay* is the time duration between two subsequent activities

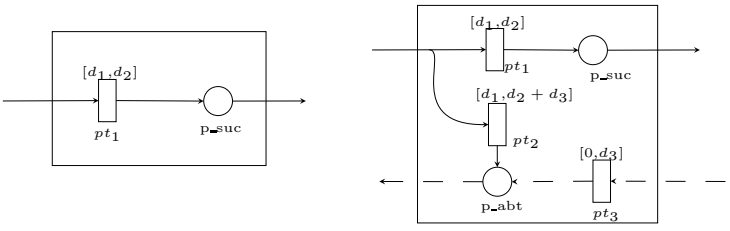

Fig. 2. Petri net representation of atomic tasks

(i.e., tasks). *Duration* is the maximum time required to finish a task. *Compensation duration* refers to the maximum time required to compensate a failed task. Delay, duration and compensation duration are expressed by integer values.

Both traditional control flows and control flows associated with compensable transactions can be easily edited and displayed graphically in the editor. The modeling elements are displayed in Fig. 3, using a notation similar to many workflow modeling languages. The editor produces workflow models which are *correct by construction* [31] which essentially means that incorrect composition of workflow activities is prevented. CWML$_T$ is a structured workflow modeling language which follows constraint-based approach and for this reason it becomes possible to not only guarantee that processes run correctly regarding their control and data flow, but also regarding the validity of the specified semantic constraints. As each workflow component has an underlying Petri net structure, the language has a sound mathematical foundation.

The editor is built as an Eclipse Plugin [6] using the Eclipse Graphical Editing Framework (GEF) [7]. Because of this architecture, the NOVA Editor is available in the development platform. Application developers can create models in a Java project, and generate workflow service classes from it (see section 3). Modeling, development and verification can be done in the same Eclipse Platform.

3 The NOVA Engine

The NOVA Workflow is developed using an SOA architecture. The engine was developed on the Spring and Hibernate platforms; Spring is a widely used open source framework that provides a consistent programming and configuration model that is well understood and used by developers worldwide. From the NOVA Editor, Workflow service classes are automatically generated to be deployed in the Spring container. These services are exposed to the outside world by service provider interfaces. The lifecycle of a workflow service bean (the class that contains the business logic) is managed by the spring container and at the time of instantiation, service beans register themselves to the workflow engine.

The NOVA Engine provides two modes for workflow engine integration: i) loosely coupled integration, ii) tightly coupled integration. The one to be selected depends on the system architecture. When the application services are not

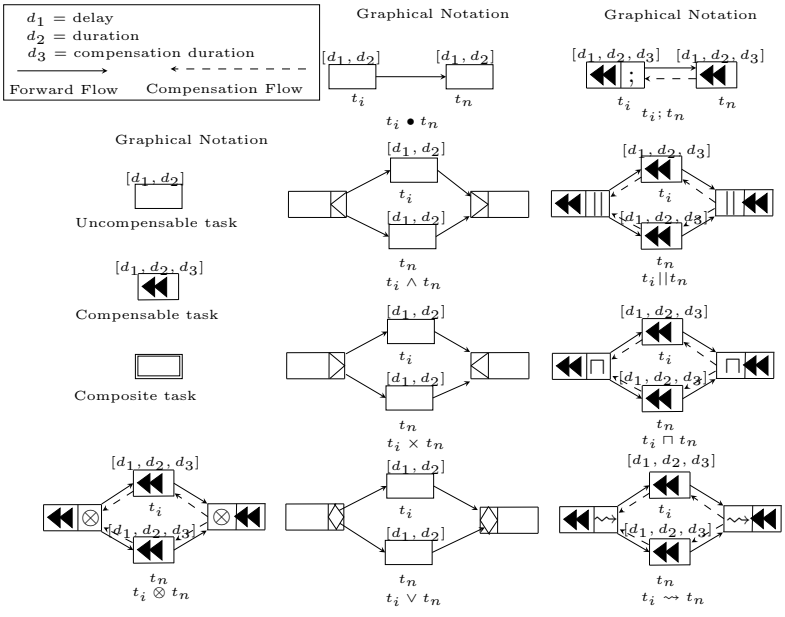

Fig. 3. Modeling Elements of CWML$_T$

deployed in the spring framework, loosely coupled integration should be selected; by this integration, workflow services are invoked by application services from outside the spring container. The NOVA Engine updates the task status when a particular service bean is invoked. On the other hand, if the application services are deployed in the spring container, tightly coupled integration is recommended; here application services are extended by workflow service classes.

Although the NOVA Workflow provides all support to develop a full swing client application, the current implementation of the NOVA Workflow does not generate default forms to take input from the user. In future, a form builder (or default form generator) will be incorporated with the NOVA Workflow. With the current version of NOVA Workflow, client applications may be developed in various platforms and they may communicate with the NOVA Engine using RMI, JSP, HTTPInvoker, WebService, etc. In future we intend to support greater mobility using sophisticated technologies such as iPads, Tablet PCs, etc. Accessibility and mobility are important for health care applications. It is anticipated that advanced mobile applications will improve outcomes as physicians, nurses and other clinicians can access both recent and historical information while visiting a patient. Advanced interoperability vis a vis international standards such as HL7, OpenEHR is on the horizon for future research.

4 The NOVA Translator

Once a workflow is designed with compensable tasks, its properties can be verified by model checkers such as SPIN, SMV or DiVinE. Modeling a workflow

with the input language of a model checker is tedious and error-prone. In [30] we provided a manual translation from CWML to DVE the input language of the DiVinE model checker, which was extended to an automated translator to CWML_T in [27]. This automated translation method has been integrated into the NOVA Workflow. Now, using the Editor one can graphically design a workflow using the CWML_T and write the business logic for the tasks (in Java); then the translator automatically translates the model to DVE (see [30,27]).

However, the time required for the verification was often unacceptable. Our experiments showed that even though DiVinE is equipped with the Partial Order Reduction technique and several different model checking algorithms, it required a great deal of memory and time for the verification of large models. In [29] we developed a model reduction algorithm for the models built using (untimed) CWML. The algorithm has been implemented in the NOVA Translator, which takes a workflow model and the specification of an LTL_{-X} property ϕ and reduces the model, based on the property ϕ. The proof of the stuttering equivalence of the original and reduced models may be found in [29]; thus the truth of ϕ is preserved and reflected. A demonstration of the effectiveness of the proposed method in reducing the size of the state space may be found in [29]. We expect the proposed algorithm can be easily applied to any block-structured modeling language (e.g., ADEPT2 [31]), and currently we are extending the reduction to models built using CWML_T. The reduction algorithm incorporates the feature of data-awareness currently found in many workflow modeling languages. Other reduction algorithms involving time and including the notion of *leaping time* [37] are currently under investigation. An extention of the translator to other model checkers, i.e., SMV, SPIN, Maude are under development which will allow us to provide an in-depth performance comparison.

5 The NOVA Monitor

Workflow monitoring is an active research area which has great importance especially in safety critical systems for enforcing policies, and achieving effciency and reliability goals. Monitoring is a frequent requirement in healthcare environments where monitor systems are typically configured to notify clinicians about abnormalities in a process. Designing a monitor for a time constrained compensable workflow in a healthcare setting is complex and it must be verified to ensure it operates properly before its use with a patient.

The NOVA Monitor uses a graphical modeling language for monitoring workflows which is based on Time Petri nets [14], and integrates such time constrained monitors (called WMon-nets) with workflow models (called CWF-nets) built using CWML_T. Fig. 4 shows the graphical notation for workflow monitor components. A green transition is associated with the forward transition (pt_1 in Fig. 2) of an atomic (uncompensable or compensable) task in a CWF-net,

and a yellow transition is associated with the compensation transitions (pt_2, pt_3 in Fig. 2) of an atomic compensable task in a CWF-net.

Green and yellow transitions are virtual transitions and do not execute like ordinary transitions in a Petri net. It is through these virtual transitions that we integrate a WMon-net with a CWF-net. The output places of green transitions and yellow transitions get tokens when the corresponding transitions (pt_1, pt_2, pt_3) execute in a CWF-net (see Fig. 5).

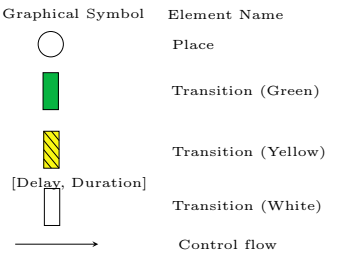

Fig. 4. Graphical representation of monitor components

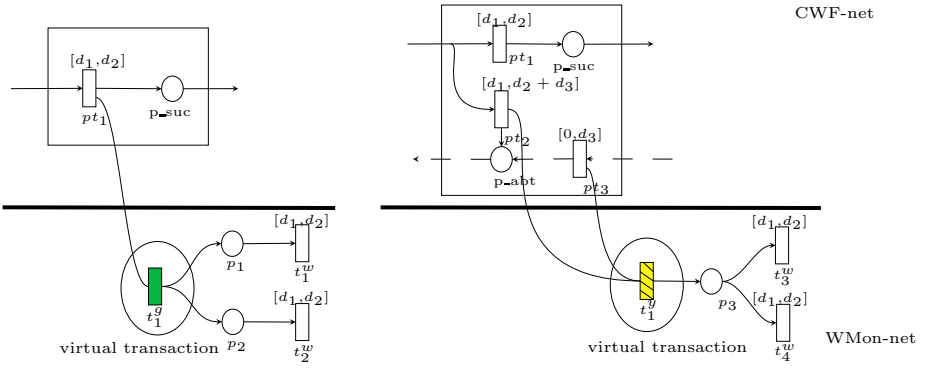

Fig. 5. A virtual transition

In order to verify properties of a monitor (modeled as a WMon-net) we combine the state space of the monitor with the state space of the workflow. A monitor monitors the processes of a workflow so it cannnot generate its own state space without input from that workflow. From the generated state space, the model checker can verify the properties of the system. In the case study, we show a compensable workflow with a monitor and verify properties of the workflow including properties to show that the monitor is working correctly. An ontology was incorporated with the monitor; results of complex queries to the ontology can guide the control flow. It is anticipated that the workflow monitor may be integrated with other workflow languages, with Petri net based foundations. Details of the monitor system may be found in [4].

6 The NOVA Browser

Usability is an important quality in any product but is often neglected in software products. It was found that EHR (Electronic Health Record) usability is at the root of many medical errors [21]. For healthcare systems, usability is not an expectation; rather it is an essential requirement, as patient safety is involved

and clinician's time is a valuable resource. We now present a user friendly browser, called the NOVA Browser, for EHR with which health professionals can view a patient's medical information without losing the patient's context while browsing. The flexibility and capacity for brainstorming enhances the user experience. The browser also incorporates a time travel view and a chart view that helps health professionals observe and monitor a patient's medical condition. The unique time travel view of the browser makes it a helpful tool for cause and effect analysis.

6.1 Current Features of the NOVA Browser

Hierarchical Representation of Data. The NOVA Browser hierarchically represents a patient's EHR in a mind map. A mind map [17] is a graphical way to represent ideas and concepts. The nodes (i.e., ideas, concepts, items, etc) are represented hierarchically in a mind map. A mind map (as opposed to traditional notes or text) structures information in a way that resembles much more closely how the brain actually works. Since it is an activity that is both analytical and artistic, it engages the human brain in a rich way, helping in all its cognitive functions. While pictorial methods for recording knowledge and modeling systems have been used for centuries in learning, brainstorming, memory, visual thinking, and problem solving by educators, engineers, psychologists, and others, its use in software systems to provide a means to involve the user more with the system is rare (e.g., mind map has been used in OpenEHR as an archetype [1]).

Fig. 6 shows an EHR representation in the NOVA Browser. The browser displays the records of a particular patient's case. The centre of the map shows the case number (Case-1). The clinician can unfold any of the branches and can view the details of the case. The browser performs a database query using the case number and loads the records from database tables or views. The NOVA Browser provides a case selection window with which the clinician can switch to a different patient's case.

Time Travel View. In order to enable the time travel view, the database tables include the Timed Table, to preserve the historical information and time. In a Timed Table, no records are deleted or updated by replacing the original record; instead of an update operation to a row, a new entry with status UPDATE is inserted into the table and a column is used to indicate the parent record, whose information is being updated. When displaying the records in the browser, only the latest records are shown. When the user travels back, the browser fetches historical records and displays them in the map.

The time travel view provides an easy way for clinicians to go back to when a certain record was inserted or updated and then check its effect by travelling forward from that time. The browser provides four types of time travel: Travel Backward (or Travel Forward) to a past (or future) time when a selected record was inserted, updated or deleted (in this case the user needs to select a node), and Travel One Step Backward (or Forward), which is travel to the previous

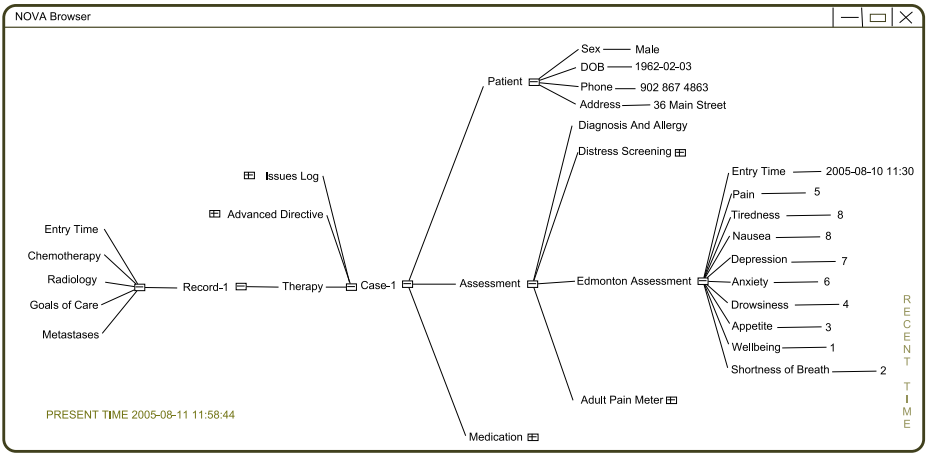

Fig. 6. Hierarchical Data representation in NOVA Browser

(or next) time that a record was inserted or updated for the selected case (in this case the user does not select a node and the search operation is performed globally on all tables for the case).

If the clinician selects the 'Assessment' node and travels back, the browser will jump to the time when an Assessment record was inserted or updated. Note that Assessment is an abstract base class with three concrete subclasses (i.e., Distress Screening, Edmonton Assessment, Adult Pain Meter). The NOVA browser uses hibernate [9] to communicate with the database. As hibernate is an Object Relational Mapper, any query performed on a base class is executed on all its subclasses. The NOVA Browser displays a transition from one map to another by doing animation about the Z-axis, which gives the impression of travelling backward or forward.

Chart View. Charts and graphs play an important role for analyzing information. Detecting patterns in patient populations is an esential function in the administration of health care systems. The visual representation of complex information can help process large amounts of data to detect and observe such patterns. However it is very difficult to pre-configure all the charts with all the different combinations of parameters which may be needed by clinicians and administrators. The NOVA Browser incorporates a chart view allowing the user to select the desired chart parameters. The clinician selects some nodes from the browser, adds them to the parameter list and then selects a time range. The chart viewer generates the chart using those selected parameters by plotting time on the X-axis and the parameters on the Y-axis. As a clinician can select any node from the browser, the browser will either plot the exact values of the parameter or present them symbolically.

6.2 Proposed Application of the NOVA Browser

We propose applying the NOVA Browser to process diagnosis. To provide flexibility to the system, we allow two ways of interacting: i) Workflow based, ii) Task based. The workflow system will provide a worklist window from which the user can select a task and execute it. Alternatively, the user may skip the workflow and can fill in and submit a form related to a task. A workflow provides better support but is less flexible. We will allow the user to skip the workflow, as initially all patient case scenarios are not known so the workflow should not restrict the user to perform an emergency job. The workflow may be adjusted later on through a redesign.

All the activities performed by the user are recorded in an event log which will be used to restructure the workflow. Two types of events may be found in the event log: i) a workflow event for a task being executed from the worklist, and ii) an ad-hoc event for a form being executed in an ad-hoc manner. We have addressed the issue of an evolutionary process restructuring through a process diagnosis mechanism. We propose a new design of process diagnosis in the NOVA Workflow which incorporates a process mining technique using the NOVA Browser.

Motivation of the Work. The term process mining refers to methods for distilling a structured process description from a set of real executions. In [20], Cook and Wolf described three methods for process discovery: one using neural networks, one using a purely algorithmic approach, and one using a Markovian approach. The authors considered the latter two as the most promising. The purely algorithmic approach builds a finite state machine (FSM) where states are fused if their futures (in terms of possible behavior in the next k steps) are identical. The Markovian approach uses a mixture of algorithmic and statistical methods and is able to deal with noise. However, they did not provide an approach to generate explicit process models. The idea of applying process mining in the context of workflow management was first introduced in [12]. This work is based on workflow graphs, which are inspired by workflow products such as IBM MQSeries workflow (formerly known as Flowmark) and InConcert.

Van der Aalst et. al., studied a number of process mining or workflow mining techniques in [36] and pointed out two problems. The first is to find a workflow graph generating events appearing in a given event log and the second is to find the definitions of edge conditions (i.e., pre-conditions). In [36], they provided a concrete algorithm for tackling the first problem.

For a simple system with a few tasks and enough workflow logs it is quite easy to construct a process model, but for more realistic situations (e.g., healthcare) there are a number of complicating factors [36]:

1. Mining is difficult in large workflow models and has a high degree of complexity if the model exhibits alternative and parallel routing (in which case the workflow log will typically not contain all possible combinations).

2. Workflow logs will typically contain noise, i.e., parts of the log may be incorrect, incomplete, or refer to exceptions. For example, events can be logged incorrectly because of human or technical errors.
3. Events can also refer to rare or undesirable events. Consider, for example, a workflow in a hospital. If, due to time pressure, the order of two events (e.g., make X-ray and remove drain) is reversed, this does not imply that this would be part of the regular medical protocol and should be supported by the hospital's workflow system. Also two causally unrelated events (e.g., take blood sample and death of patient) may happen in a sequence without implying a causal relation. Exceptions which are recorded only once should not automatically become part of the regular workflow.

Process Diagnosis Using the NOVA Browser. For a safety critical system it is not desirable to deploy the workflow model discovered by the automated process mining tool without a proper validation by domain experts for the workflow model discovered by process mining. The conditions of *XOR*, *OR*, *Loop* discovered by a process mining tool from the event log may not be the exact condition for the selection of the branches. For a healthcare application there are numerous parameters that guide the flow of the tasks.

The proposed workflow management life cycle consists of 6 steps (i.e., 1. Workflow Design, 2. Workflow Validation, 3. Workflow Enactment 4. Event Log 5. Process Mining 6. Process Restructure), where step 5, 6 and 2 are part of the process diagnosis. Consider the workflow fragment 'Medicine Administration' as shown in Fig. 7. The initial design of the Loop condition was to *iterate until the dose is finished*. During the execution of the workflow it was found that a patient was allergic to certain drug and the 'Medicine Administration' task was cancelled for this reason.

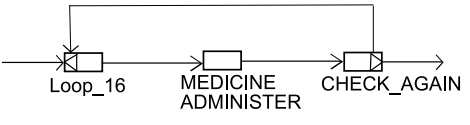

Fig. 7. Medicine Administration

Let us assume that the allergy information was inserted into the system and another change in the patient's medical condition happened at the same time; for example, the patient's PPS (Pulse Per Second) value reduced. Here the decrement of the patient's PPS value is noise which makes it hard to identify the exact Loop condition for a process mining tool.

We propose a design for a 'Process diagnosis tool' using the NOVA Browser to handle these problems. The proposed system is shown using the NOVA Editor in Fig. 8, where the workflow model discovered by the process mining tool can be restructured by the user easily and efficiently. The workflow component's view is shown at the bottom left side of Fig. 8. The user makes some changes to the workflow model, and invokes an analyze action to the system; by analyzing the

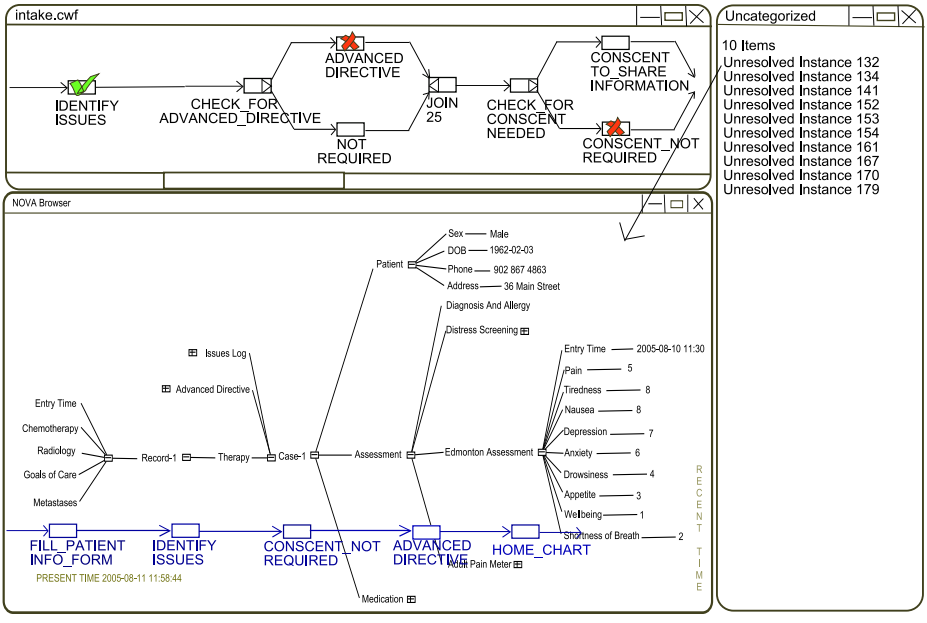

Fig. 8. Process diagnosis using the NOVA Browser

event logs, the system searches for the workflow instances which do not satisfy the redesigned workflow model. The unresolved workflow instances are shown at the top right side of the editor. The user opens an unresolved workflow instance in the NOVA Browser (shown in the middle of the editor) to analyze the case. The sequence of the tasks' execution for the selected case is shown at the bottom of the NOVA Browser. The user can use the time travel view in the Browser to go back to past records and see under what condition the tasks were executed. For example, consider the above 'Medicine Administration' problem; the user can go back in time to when the drug administration was stopped, find the allergy recorded at that time and then edit the workflow and restructure it with the exact condition. With such a mining and reconfiguring feature, a WfMS may be considered to be adaptive.

7 Case Study

We model a workflow and a monitor system following the guidelines for the management of cancer related pain in adults [16]. If a patient is taking a strong opioid, a pain reassessment should be done after a certain interval. The guideline for the strong opioid regimen says that if a patient is responding (i.e., current pain level is less than previous pain level) then another reassessment should be done within a week; if a patient is not responding then it suggests a different reassessment interval depending on the current pain level. The guideline suggests 'Opioid toxicity' or the 'Continuation of dose titration' depending on the 'Response'. 'Management of side effects' is a compensation for these processes.

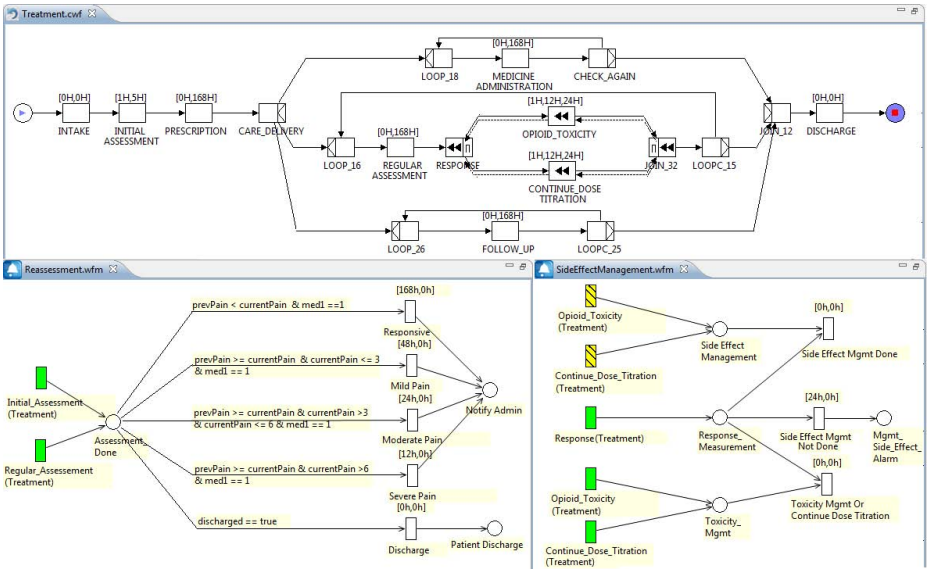

Fig. 9. "Treatment workflow", "Reassessment monitor", and "Side effect management monitor" for patients

Fig. 9 shows the treatment workflow (a time constrained CWF-net) at the top, the reassessment monitor (a WMon-net) at the bottom left which integrates with both the INITIAL_ASSESSMENT and REGULAR_ASSESSMENT tasks, and the side effect management monitor (a WMon-net) at the bottom right which integrates with the compensable tasks OPIOID_TOXICITY and CONTINUE_DOSE_TITRATION. In this model, when a patient is admitted (task INTAKE is executed) to the hospital an initial assessment is done (task INITIAL_ASSESSMENT is executed) where the patient's pain level is recorded. A physician prescribes medicine for the patient (related task is PRESCRIPTION) which may be updated at Follow up. While the patient is taking his medicine (task MEDICINE_ADMINISTRATION executes), a regular assessment is done (task REGULAR_ASSESSMENT is executed) after certain interval concurrently with other processes, e.g., Medicine Administration, Follow up. Note that, CARE_DELIVERY is an *AND-Split* task which activates all of its outgoing branches and RESPONSE is an *Internal choice split* task which activates one of its outgoing branches during execution. OPIOID_TOXICITY and CONTINUE_DOSE_TITRATION are compensable tasks and they compensate for any side effects found during execution.

The monitor system observes the interval of assessment and notifies the clinician if another reassessment is not done within the time suggested by the guidelines. The general knowledge base for medicine is very large, and frequently organized as a medication ontology. To show how our system can integrate with an ontology, we designed a small ontology in OWL 2.0 representing the facts and rules about strong opioids used for querying reassessment time. We integrated

the ontology with the monitor system using the FaCT++ reasoner. The system generates a query with the list of medication that a patient is taking, and the current and previous pain levels as parameters and sends them to the FaCT++ reasoner. The reasoner computes whether the medicine is a strong opioid and returns the suggested reassessment time. Some of the properties we verified are provided below with their LTL formulas:

- If a patient is under the strong opioid regimen, not responding (to medication), current pain level is > 6 and another reassessment is not done within 12 hours, the clinician will get a notification (In LTL, G ((strong_opioid_patient && patient_is_not_responsive && cur_p_level_gr_than_six && reassessed_twelve_hrs_ago) $->$ F (clinician_is_notified))).
- If a patient is discharged, the clinician will not receive any reassessment alert (In LTL, G (patient_discharged $->$! F (clinician_is_notified))).
- If required 'Side effects' are not managed within 24 hours, the clinician will get a notification (In LTL, G (response_measured && side_effect_not_managed && resp_measured_24_hrs_ago $->$ mgmt_side_effect_alarm))

The results of the model checking showed that first two properties were false, and provided counter examples. For the first property the counter example says if those conditions are true then the clinician may not get a notification if the patient is discharged. For the second property, the counter example says the clinician receives the notification if the discharge operation executes at the same time as a notification was supposed to be sent. It is clear from the counter examples that there exists a flaw in the models; we determined that the patient's discharge was not taken into consideration in the pre-conditions of *Responsive*, *Mild Pain*, *Moderate Pain* and *Severe Pain* transitions. As a result while the *Discharge* transition is ready, other transitions could possibly be ready and execute. The initial model was corrected by rewriting the pre-conditions and subsequent model checking showed that both properties were satisfied. Due to space limitation we presented a small case study here which involves the modeling of both a timed Compensable Workflow, with monitors and the translation to DVE and model checking. The workflow was executed in a J2EE server and interfacing to the Ontology was done by the FACT++ reasoner. Altogether, we used the NOVA Editor, Engine, Translator and Monitor. Interested readers are referred to [29] where they will find details of a much larger case study involving the reduction.

8 Related Work

Petri nets [28] is a popular formalism for the design of concurrent systems because of its sound mathematical foundations. Many analysis techniques are available for Petri nets. Workflows may be designed by Petri net tools such as TINA [15], Romeo [33], etc. Reo [24] is a graphical channel-based coordination

language that enables the modeling of complex behavioral protocols using a small set of channel types with well-defined behavior. Designing a large workflow model with these tools ([15,33,24]) is difficult to manage. Designing a workflow with compensation using these tools is particularly complex as the model becomes very large; the use of a high level modeling language is preferable. These tools, moreover do not use reduction techniques explicitly to verify a large model. In [22] the authors provided a reduction technique for Coloured Petri nets. These techniques preserve the liveness of the model and any LTL formula that does not observe the reduced transitions of the net. This approach to reduction differs from ours [29]; in our approach we take the model (M) and the property (ϕ) both into consideration and reduce the model in such a way that the reduced model, $M' \vDash \phi$ *iff* the original model $M \vDash \phi$. Another difference between [22] and our approach is merging vs. reducing transitions. We follow the later approach which reduces tasks, pre-conditions, actions and variables from the original model. It will significantly reduce the memory size for each state; as a result the memory size of the whole state space becomes less.

UPPAAL [13] is a popular timed automata model checker but the distributed version of UPPAAL is under development. The data aware verification method we presented here using the parallel distributed model checker DiVinE provides excellent support to verify large systems.

Some popular workflow management systems are YAWL [35], ADEPT2 [31], BPEL [2], etc. For workflow modeling YAWL uses workflow patterns but is an unstructured language; on the other hand, ADEPT2 uses a block-structured language. The use of structured vs. unstructured workflow language is debatable; usually unstructured workflow languages are more expressive than block-structured languages but the soundness is not guaranteed by construction as in a block-structured language. CWML_T is a block-structured language and hence the soundness is guaranteed by construction (proof in [30]). YAWL comes with limited forms of verification (e.g., livelock, deadlock, etc). In [38] the authors provided reduction rules for YAWL workflows with *Cancellation* regions and *OR-joins* to reduce the size of the workflow, while preserving its essential properties with respect to a particular analysis problem. Here the authors only focused on the soundness analysis, whereas our reduction method works for any LTL_{-X} property (The subset of LTL formulas not containing the X operator).

An ADEPT2 workflow can be verified using the SeaFlows compliance checker [23]. In [23] the authors discussed an abstraction approach which can serve as a preprocessing step; this is an efficient way to deal with the state explosion problem. This strategy is orthogonal to our strategy; our methods can be further improved by the automated abstraction technique to reduce the data domain for variables and their method can be improved by the reduction technique we discuss here.

YAWL and ADEPT2 do not have compensation mechanisms, whereas BPEL has a built-in compensation for atomic processes (there are no compensable operators). BPEL has been used in the industry for some time and there are many publicly available tools (e.g., BPEL2PN [3], WSEngineer [11]) to analyze workflows designed in BPEL. BPEL2PN does not provide data aware

verification and WSEngineer does not use any explicit reduction technique. Although BPEL has a built-in compensation, it is not t-calculus based. CWML$_T$ is more expressive and gives us flexibility in designing compensable workflow with its rich semantics.

Simmonds et al. presented the tool RuMoR in [32] which performs monitoring of web service applications, and, when violations are discovered, automatically proposes and ranks recovery plans. Properties, specified using property patterns, are transformed into finite state automata. While runtime monitoring, some compensation mechanism, verification are common to our system there are differences. RuMoR takes a BPEL program as input and translates to a labeled transition model using WS-Engineer. Monitors are specified as finite-state automata. Although data aware verification can be done in RuMoR it has limitations with respect to time. RuMoR was implemented within the IBM WebSphere using the interception mechanism, whereas the architecture of NOVA Workflow is light-weight as it uses Spring and aspect oriented programming techniques which enabling its use with various J2EE application servers. In addition, NOVA Workflow uses an ontology for decision support.

PROM [8] is an automated process mining tool which requires enough event logs to extract process flows; here in this paper we proposed another process mining tool which is semi-automated; the proposed tool is designed to be user friendly as it incorporates the NOVA Browser.

Declare [34] is a prototype of a workflow management system which follows a declarative approach to business process modeling and execution. Unlike conventional systems, which use graph-like modeling languages (e.g., Petri-nets), Declare uses logic (i.e., LTL) to model and execute business processes. Modeling time, compensation and monitors in Declare is complex and its formal verification needs further research. Moreover, many features of NOVA Workflow (e.g., runtime monitoring, verification, browser, process diagnosis, etc) are not covered in Declare.

NOVA, meaning 'new' in Latin, summarizes the four important features of the framework: compeNsation, Ontology, Verification and Adaptability. We have developed an interface to permit the workflow engine to consult an Ontology knowledge base to guide the execution of the workflow engine. This paper also presents a new technology for analyzing and visualizing healthcare information that will significantly improve the usability of EHR systems. In future we will interface the system with various laboratory and radiology equipment to display a patient's reports in the browser and develop an automated form builder. We will further support workflow flexibility for instance, by incorporating a sophisticated client application.

Acknowledgments. We acknowledge support from the Atlantic Canada Opportunities Agencies, the Natural Sciences and Engineering Research Council of Canada, the Canadian Foundation for Innovation, and the Nova Scotia Research and Innovation Trust. We are grateful for the enthusiastic support of clinicians and administrators from Guysborough Antigonish Strait Health Authority (GASHA).

References

1. OPENEHR, http://www.openehr.org/home.html (last accessed, July 2011)
2. IBM, BEA, Microsoft, SAP, Siebel. Business process execution language for web services version 1.1 (May 2003)
3. BPEL2PN, http://www2.informatik.hu-berlin.de/top/bpel2pn/ (last accessed, April 2011)
4. Rabbi, F., Mashiyat, A., MacCaull, W.: Model checking workflow monitors and its application to a pain management process. In: Proceedings of 1st International Symposium on Foundations of Health Information Engineering and Systems, FHIES 2011, Johannesburg, South Africa, pp. 110–127 (2011)
5. DiVinE project, http://divine.fi.muni.cz/ (last accessed, October 2011)
6. Eclipse plugin, http://www.eclipse.org/articles/article-plug-in-architecture/plugin_architecture.html/ (last accessed, April 2011)
7. Graphical Editing Framework, http://www.eclipse.org/gef/ (last accessed, April 2011)
8. PROM, http://www.processmining.org/ (last accessed, May 2011)
9. Relational persistence for java and .net, http://hibernate.net/ (last accessed, May 2011)
10. Spring framework, http://www.springsource.org/ (last accessed, May 2011)
11. WSENGINEER, http://www.doc.ic.ac.uk/ltsa/eclipse/wsengineer/ (last accessed, July 2011)
12. Agrawal, R., Gunopulos, D., Leymann, F.: Mining Process Models from Workflow Logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
13. Bengtsson, J., Larsen, K.G., Larsson, F., Pettersson, P., Yi, W.: Uppaal — a Tool Suite for Automatic Verification of Real–Time Systems. In: Alur, R., Sontag, E.D., Henzinger, T.A. (eds.) HS 1995. LNCS, vol. 1066, pp. 232–243. Springer, Heidelberg (1996)
14. Berthomieu, B., Diaz, M.: Modeling and verification of time dependent systems using time Petri nets. IEEE Trans. Softw. Eng. 17, 259–273 (1991)
15. Berthomieu, B., Vernadat, F.: Time petri nets analysis with Tina. In: QEST, pp. 123–124 (2006)
16. Broadfield, L., Banerjee, S., Jewers, H., Pollett, A., Simpson, J.: Guidelines for the management of cancer-related pain in adults. Supportive Care Cancer Site Team, Cancer Care Nova Scotia (2005)
17. Buzan, T.: The Mind Map Book. Penguin Books (1996)
18. Clarke, L.A., Chen, Y., Avrunin, G.S., Chen, B., Cobleigh, R.L., Frederick, K., Henneman, E.A., Osterweil, L.J.: Process programming to support medical safety: A case study on blood transfusion. In: ISPW 2005, pp. 347–359 (2005)
19. Combi, C., Posenato, R.: Controllability in Temporal Conceptual Workflow Schemata. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 64–79. Springer, Heidelberg (2009)
20. Cook, J.E., Wolf, A.L.: Discovering models of software processes from event-based data. ACM Trans. Softw. Eng. Methodol. 7, 215–249 (1998)
21. Edwards, P.J., Moloney, K.P., Jacko, J.A., Sainfort, F.: Evaluating usability of a commercial electronic health record: A case study. Int. J. Hum.-Comput. Stud. 66, 718–728 (2008)

22. Evangelista, S., Haddad, S., Pradat-Peyre, J.-F.: Syntactical Colored Petri Nets Reductions. In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 202–216. Springer, Heidelberg (2005)
23. Knuplesch, D., Ly, L.T., Rinderle-Ma, S., Pfeifer, H., Dadam, P.: On Enabling Data-Aware Compliance Checking of Business Process Models. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 332–346. Springer, Heidelberg (2010)
24. Kokash, N., Krause, C., de Vink, E.P.: Time and data-aware analysis of graphical service models in Reo. In: Proceedings of the 2010 8th IEEE International Conference on Software Engineering and Formal Methods, SEFM 2010, pp. 125–134. IEEE Computer Society, Washington, DC (2010)
25. Li, J., Zhu, H., He, J.: Specifying and Verifying Web Transactions. In: Suzuki, K., Higashino, T., Yasumoto, K., El-Fakih, K. (eds.) FORTE 2008. LNCS, vol. 5048, pp. 149–168. Springer, Heidelberg (2008)
26. Li, W., Fan, Y.: A time management method in workflow management system. In: Grid and Pervasive Computing Conference, pp. 3–10 (2009)
27. Mashiyat, A.S., Rabbi, F., MacCaull, W.: Modeling and Verifying Timed Compensable Workflows and an Application to Health Care. In: Salaün, G., Schätz, B. (eds.) FMICS 2011. LNCS, vol. 6959, pp. 244–259. Springer, Heidelberg (2011)
28. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (1989)
29. Rabbi, F.: Design, development and verification of a compensable workflow modeling language. MSc Thesis, St. Francis Xavier University (2011)
30. Rabbi, F., Wang, H., MacCaull, W.: Compensable WorkFlow Nets. In: Dong, J.S., Zhu, H. (eds.) ICFEM 2010. LNCS, vol. 6447, pp. 122–137. Springer, Heidelberg (2010)
31. Reichert, M., Rinderle, S., Kreher, U., Acker, H., Lauer, M., Dadam, P.: ADEPT2 - next generation process management technology. In: Proceedings Fourth Heidelberg Innovation Forum, Aachen, D.punkt Verlag (April 2007)
32. Simmonds, J., Ben-David, S., Chechik, M.: Guided recovery for web service applications. In: SIGSOFT FSE, pp. 247–256 (2010)
33. Traonouez, L.-M., Lime, D., Roux, O.: Parametric model-checking of stopwatch petri nets 15(17), 3273–3304 (2009)
34. van der Aalst, W., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science - Research and Development 23, 99–113 (2009)
35. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: yet another workflow language. Information Systems 30(4), 245–275 (2005)
36. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: a survey of issues and approaches. Data Knowl. Eng. 47, 237–267 (2003)
37. Wang, H., MacCaull, W.: An efficient explicit-time description method for timed model checking. In: PDMC, pp. 77–91 (2009)
38. Wynn, M.T., Verbeek, H.M.W.E., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Reduction rules for YAWL workflows with cancellation regions and OR-joins. Information and Software Technology 51(6), 1010–1020 (2009)