Sensors & Transducers, Vol. 175, Issue 7, July 2014, pp. 198-206

Sensors & Transducers

© 2014 by IFSA Publishing, S. L.
http://www.sensorsportal.com

A Context-aware Workflow Framework and Modeling Language

Pengfei WANG, * Huifang LI, Baihai ZHANG
School of Automation, Beijing Institute of Technology,
5 South Zhongguancun Street, Haidian District, Beijing, 100081, China
Tel.: +86-10-68914352
E-mail: 10906034@bit.edu.cn, wpf999@yeah.net

Received: 6 May 2014 /Accepted: 30 June 2014 /Published: 31 July 2014

Abstract: In the pervasive and mobile computing environment, workflow and context information are closely linked, workflow management system have to interact with a variety of sensors. Therefore, the design and development context-aware workflow applications become complex and difficult to migrate to other platforms. In order to simplify the development of context-aware workflow applications and enhance its portability, a new context-aware workflow framework is proposed in this paper. This framework introduced into the context-aware middleware. Context-aware middleware shielding the various bottom sensors work details for the application and make the workflow applications focus on using high-level context, so as to realize the intelligent workflow management. This framework also introduced into ontology based modeling language for context-aware workflow. Case study shows that our proposed modeling language has good adaptability, and can be used to easily describe any sophisticated context-aware workflows. *Copyright © 2014 IFSA Publishing, S. L.*

Keywords: Context-aware computing, Context-aware workflow, Framework, Modeling language, Web ontology language.

1. Introduction

In the context-aware computing fields, context is any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves [1]. Context information may include, but are not limited to location, time, user status, network bandwidth, weather conditions. A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task [1]. In other words, a context-aware application can adapt its behaviors according to the context that is gathered from various types of sensors.

Workflow is automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [2]. Workflow technology has been widely used in office automation system in past decades, but this powerful technology cannot adapt some new fields such as mobile Internet, smart factory, smart home, because the traditional workflow management system did not give full consideration to the integration of context information that is essential to the new fields. In order to manage and control context sensitive business process, context-aware workflow management system is needed.

Context-aware workflow management system (CWfMS) is based on context-aware computing, and integrates sensor networks and context-aware middleware. CWfMS can perceive, identify, analyze, understand contextual information, and execute workflow intelligently based on context.

Due to inherent complexity of context-aware applications, CWfMS needs to access various types of sensors, and handles a variety of complex context information. Therefore establishing a good system framework is the key to successfully build CWfMS. Although some context-aware workflow management system frameworks have been proposed [3-10], but they only considered the design used in specific areas, thus there is no a generic framework for context-aware workflow management system.

In order to solve the above problems and simplify CWfMS design and development, we propose a general framework of CWfMS based on middleware. Different from traditional workflow management system, CWfMS includes context-aware platform that utilizes middleware to shield differences of sensors and achieves induction and fusion from low-level context to high-level context, providing uniform context information interface for workflow engine.

The rest of the paper is organized as follows: Section 2 briefly presents related work of context-aware workflow management system and workflow modeling problem, Section 3 presents the framework of CWfMS which is based on context-aware computing, Section 4 propose a context-aware workflow modeling language based on OWL (Web Ontology Language), Section 5 introduces the technologies related to the implementation of CWfMS, Section 6 gives the example of proposed context-aware workflow modeling language, and Section 7 concludes the paper.

2. Related Works

There are some researches about the context-aware workflow management system. CAWE [3-5] is architecture of context-aware workflow system and it is applied in medical domain. CAWE is composed of several main components: Context-Aware Workflow Manager, Context Manager Service and device-dependent User Interface (UI). The Context-Aware Workflow Manager runs a workflow engine on the process specification which defines the business logic of the composed application. The Context Manager Service provides the contextual information during the execution of the application. The device-dependent UI component retrieves the context information needed to adapt the UI pages from the Context Manager Service.

In [6] a layered system model has been introduced, which proposed a three layered model of context-aware workflow system. There are three layers in the model: context provisioning layer (CPL), context integration layer (CIL) and smart workflow layer (SWL). The CPL is responsible for managing the context information. The CIL uses the generic interface provided by the CPL to integrate information into high-level representations and access patterns. The SWL realizes the smart workflows.

A context-aware workflow system for smart home has been proposed in [7]. The system uses contexts in a workflow service scenario as conditions of service execution, and dynamically derives service transition according to a user’s situation information generated from real environments. The architecture of system composed of context-aware scenario editor, context-aware workflow engine and context processor. The uWDL (ubiquitous Workflow Description Language) [8] is used to describe workflow service scenario in the system.

In [9, 10], a context model and a context-aware workflow management algorithm CAWM has been presented for pervasive campus navigation. Advantageous to the existing workflow management technologies, CAWM algorithm can adjust workflow execution behaviors based on current context information. CAWM is able to guide campus users intelligently and transparently. CAWM manages workflows in the event-driven way, including atomic and composite events. They defined five types of atomic events: context change event, method call event, state transition event, (absolute and relative) time event and user-defined event. The composite events are composed of the atomic events or lower-level composite events, using the composite operators with specified priorities. They also modeled the workflow management algorithm and analyzed its correctness through Petri nets.

3. The Framework of CWfMS

3.1. System Components

Fig. 1 shows our proposed framework of context-aware workflow management system. It consists of the following collaborative components:

- Sensors subsystem: Sensor in the paper means all kinds of physical and logical sensor by which Context provisioning platform can get the context information of the real world.
- Context provisioning platform subsystem: Context provisioning platform is server-side software that collects, processes and manages context information which will be used by workflow execution engine.
- Workflow execution engine: Server-side software that interprets and runs the workflow model, interacts with the workflow client GUI, calls the external applications which realize the business functions.
- Workflow execution database: to store the real-time data and history data that generated in workflow execution period.
- Workflow simulation engine: Server-side software system that implements the simulation of workflow model.

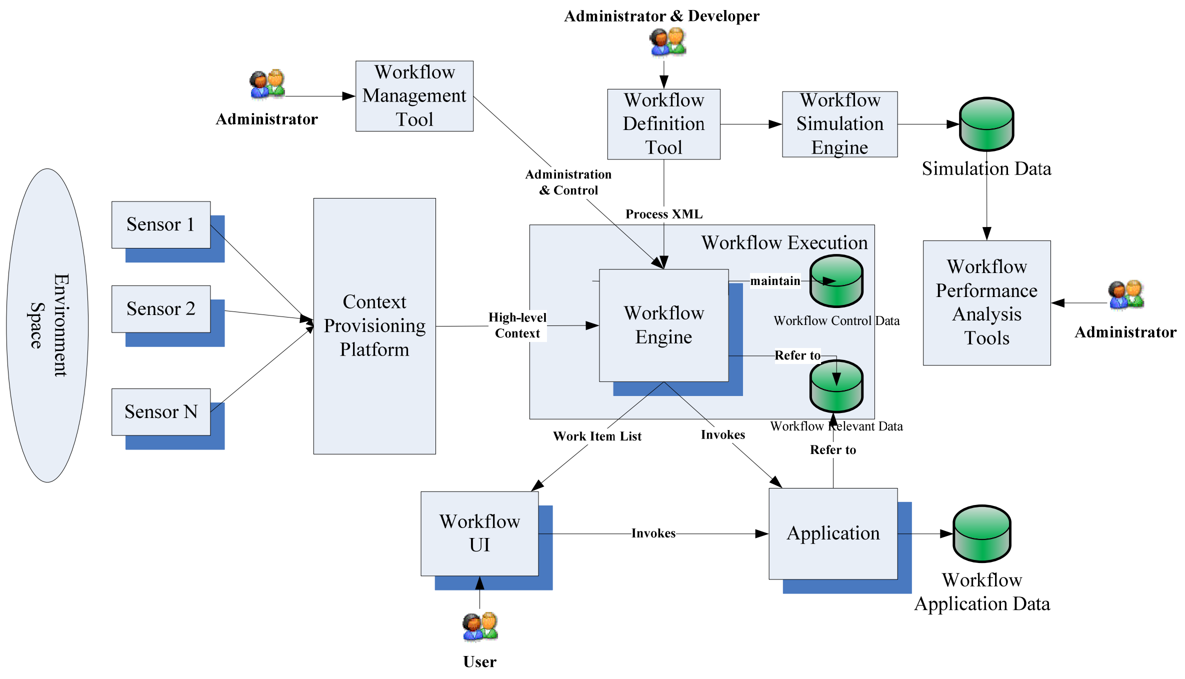


Fig. 1. The framework of CWfMS.

• Workflow simulation database: to store the huge data that is generated in workflow simulation period.

• Workflow designer: desktop software that defines workflow model which is represented by XML document and will be sent to workflow execution engine or workflow simulation engine for execution or simulation.

• Workflow administration tool: desktop or mobile software that monitors and administrates the status of workflow execution engine.

• Analysis platform: to analysis the performance of workflow. Analysis platform can provide with four type of analysis function: performance analysis based on simulation data, performance analysis based on history data, real-time analysis and synthetic analysis based on simulation and history data.

3.2. Context Provisioning Platform

The Context Provisioning Platform (CPP) is important to context-aware workflow management system, because the CPP collects, represents and processes context information. Fig. 2 shows the detail of context provisioning platform. It consists of the following components:

• Sensor access layer: It integrates all sorts of sensor drive or access interface, accesses to a variety of physical sensors and logic sensor, upward in a unified data format to send data. Sensor access layer of existence, make upper module need not care about lower sensor technical details, thus shielding of the bottom of the sensor differences, make the system have good scalability.

• Context filter: Due to various reasons (such as sensor failures, data transmission error, etc.), the original context data may be wrong or conflict situations. For example, sensor submitted indoor temperature if greater than 60 degrees Celsius, can generally considered illegal data, need to be filtered out. Context filter receiving sensor access layer send data to eliminate error data and conflict data, and then submitted to the context reasoning machine.

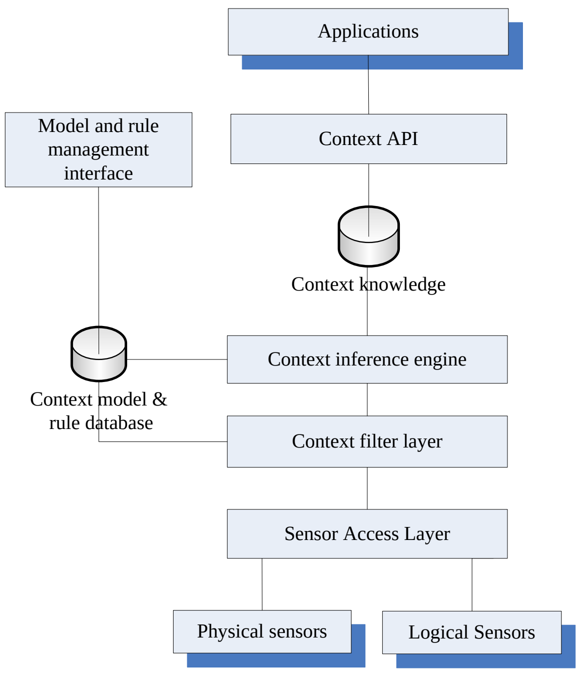


Fig. 2. Context provisioning platform.

• Inference machine: It is the core part of the CPP. According to the context model and inference rules, the inference machine performs reasoning operation, to low-level context as input and high-level context as output. New high-level context will be saved to the context knowledge database, inference required context model and inference rules also saved in the context model database. For example, in some application, the application needs to sense the user 's room, and the sensor can only provide the user's position coordinate data, the inference machine map position coordinate to the room number, position coordinates is low-level context, the room number is the high-level context.

• Context model database: to store inference rules and context models that are represented by ontology model language.

• Context knowledge database: to store low-level context and high-level context. High-level context can be used by workflow execution engine and workflow client GUI.

• Context application program interface (Context API): a set of API that is called by external applications which often use context to make a decision or trigger an event. Workflow execution engine gets interested in context in-formation by calling context API. There are two modes for calling Context API: context query and context event. For context query, workflow execution engine submitted a query to the context API, which query context knowledge database and return the result to the workflow execution engine. For context event, since the context API scan the context knowledge database, found to be subscribed to upper context update (i.e. the occurrence of a particular context events), will notify workflow execution engine in callback function.

3.3. Using Context in Workflow

In CWfMS, there are 4 kinds of schemas of context information used in context-aware workflow.

Schema 1: Transition condition. It is a basic working mode. When a workflow instance is executing, workflow engine that uses context to control workflow routing determines workflow transition according to the user location, network bandwidth, user's social state, etc.

Schema 2: Workflow resource allocation. In a task node of the workflow, the actor is called resource (operator, application, machinery and equipment, etc). The status of resource is considered as context. The performance of workflow management system can be greatly enhanced according to the context based resource allocation. For instance, resource can be allocated according to the operator's state (idle/busy) and the context-aware system evaluates the operator's busyness degree according to the related sensors in intelligent space and the workflow engine pushes tasks according to the degree, and the tasks can be reallocated to the idle operators.

Schema 3: Workflow instance starting. When an event occurs in the context, workflow instance is started according to the predefined rules. For example, in a smart warehouse, when the stock of one instrument falls below its threshold, then the context-aware system will trigger a corresponding inventory inefficiency event. Driven by this event, workflow engine will automatically start a purchase process and notify related staff to purchase the instruments.

Schema 4: Workflow exception handling. Workflow exception means that activity failure and/or workflow failure caused by outer exception of workflow, system exception of workflow, and activity execution exception, which causes the workflow does not work normally according to the predefined workflow model. Workflow management system can handle the exceptions when the process is executing according to context, thereby enhances its exception handling ability of workflow.

4. Context-aware Workflow Model Language

4.1. OWL Language

OWL (Web Ontology Language) [11] is an ontology modeling language recommended by W3C. OWL language has the following characteristics:

1) Major language elements of OWL language are class, property, individual among which property is categorized into object property and datatype property.

2) Class in OWL that is essentially a set concept, is different from the class in the program design languages. Class of OWL includes subClass and superClass and any class is its own subclass and superclass. In fact, we can consider subclass as subset and superclass as superset. In OWL, two classes, Thing and Nothing, are predefined. The thing class can be seen as a universal set and nothing can be seen as empty set.

3) In OWL, the concept property is also different from that of programming language. The object property in OWL is a binary relation over two sets, but "datatype property" equals to the concept property in programming language.

4) Property domain and range (which can be seemed as two classes) of an object in OWL can be user defined.

5) Datatype property in OWL associates an object with its data.

6) Individual in OWL equals to the elements in a set.

4.2. Advantages of Modeling with OWL

1) OWL can fully describe and express context information in workflow. Context is complex and

includes many types of information such as computing context, user context, physical context, time context and social context, etc., so it seems better to use OWL to model the context and relationships.

2) OWL language is an international standard, and is hardware, operation system and programming language independent that can be interchanged and shared among different software systems.

3) OWL language supports logic reasoning that becomes especially important when modeling. Context-aware server or context-aware middleware need to collect, filter, aggregate and reason over the original data, so using OWL to model context can be convenient to inference machine.

4.3. Meta-model of Context-Aware Workflow

Workflow meta-model is the model that defines the workflow semantic model constructs and rules. In this paper, we use OWL to model context-aware workflow, action nodes and router nodes, actor, execution time, place and other contexts. These modeling elements are defined as classes in OWL language using inheritance (each class derived from its super class).

Class Node is defined as the superclass of all nodes, Class TaskNode is defined as the superclass of all task nodes, Class RouteNode is defined as the superclass of all route nodes. Subclasses of class Node include class StartNode, EndNode, TaskNode and RouteNode. Subclasses of class TaskNode include class ApplicationTask, HumanTask and SubWorkflow. Subclasses of class RouteNode contain class ForkBegin, ForkEnd, SwitchBegin, SwitchEnd, LoopBegin and LoopEnd.

Class Agent is defined as the activity executor in workflow and derivates subclass Application and Person denoting automatic task executor and manual executor.

We define activity execution time class as Time, and place class as Location. Two subclasses Coordinate and Place are derived from Location, denoting the location described by the geographical coordinates and place expressed with natural language respectively. All classes in the meta-model are inherited from the root class Thing, their inheritance relation is shown in Fig. 3.

4.4. Process Model Elements

Process model elements include startNode, endNode, taskNode, routerNode. StartNode denotes the beginning of a process, endNode is the ending of the process, taskNode denotes all kinds of process activities, routerNode implements the control function of the processes. Node is defined as a node class from which the other classes are derived.

The beginning node is denoted as StartNode and is defined by OWL as follows:

```
<owl:Class rdf:ID= "StartNode" >
    <rdfs:subClassOf rdf:resource= "#Node" />
</owl:Class>
```

The endNode denotes ending of the process, defined with OWL as:

```
<owl:Class rdf:ID = "EndNode" >
    <rdfs:subClassOf rdf:resource= "#Node" />
</owl:Class>
```

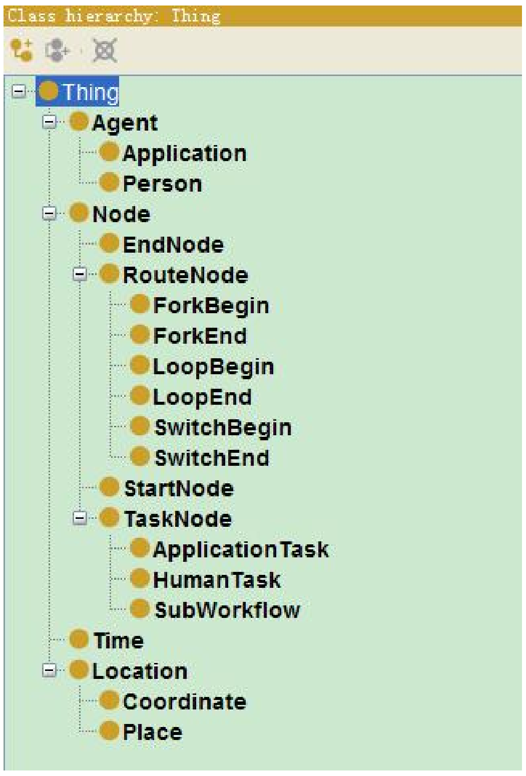


Fig. 3. Meta-model of context-aware workflow.

TaskNode denotes a task in a process. TaskNode contains three subclasses: HumanTask node, ApplicationTask node, and SubWorkflow node. Tasks are classfied into atomic tasks and compound tasks. Atomic task cannot be divided into the smaller tasks which can be categorized into human tasks and automatic tasks. Compound tasks are composed of simple/compound tasks connected by some control logic such as serial, parallel, selection, etc. Compound task is defined as a subprocess node. Subprocess node can realize modularity and reuse. Process block repeated in some processes can be defined as subprocess, so process modeller can reuse these subprocesses to enhance the modelling efficiency and avoid some repeated labors. TaskNode, ApplicationTask, and HumanTask are defined with OWL as follows:

```
<owl:Class  rdf:ID= "TaskNode" >
    <rdfs:subClassOf  rdf:resource= "#Node" />
</owl:Class>
<owl:Class  rdf:ID= "ApplicationTask" >
    <rdfs:subClassOf  rdf:resource= "#Node" />
    <rdfs:subClassOf  rdf:resource= "#TaskNode" />
```

```
</owl:Class>
<owl:Class  rdf:ID= "HumanTask" >
    <rdfs:subClassOf  rdf:resource= "#Node" />
    <rdfs:subClassOf  rdf:resource= "#TaskNode" />
</owl:Class>
<owl:Class  rdf:ID= "SubWorkflow" >
    <rdfs:subClassOf  rdf:resource= "#Node" />
    <rdfs:subClassOf  rdf:resource= "#TaskNode" />
</owl:Class>
```

RouteNode is used to denote the process logic between process nodes such as parallel, selection and loop. Parallel tasks start its execution from a parallel node and are synchronize by a command node to finish its execution. For parallel task, we use parallel start node and end node to denote such executing relationship.

Selection is a basic structure in the process. We use SwitchBegin and SwitchEnd to denote the selection.

Loop is also a basic structure in the process. We use LoopBegin and LoopEnd to denote the beginning and the end of the loop.

Definitions of RouteNode, ForkBegine, ForkEnd, SwitchBegin, SwitchEnd, LoopBegin and LoopEnd using OWL are as follows:

```
<owl:Class  rdf:ID= "RouteNode" >
    <rdfs:subClassOf  rdf:resource= "#Node" />
</owl:Class>
<owl:Class  rdf:ID= "ForkBegin" >
    <rdfs:subClassOf  rdf:resource= "#Node" />
    <rdfs:subClassOf  rdf:resource= "#RouteNode" />
</owl:Class>
<owl:Class  rdf:ID= "ForkEnd" >
    <rdfs:subClassOf  rdf:resource= "#Node" />
    <rdfs:subClassOf  rdf:resource= "#RouteNode" />
</owl:Class>
<owl:Class  rdf:ID= "SwitchBegin" >
    <rdfs:subClassOf  rdf:resource= "#Node" />
    <rdfs:subClassOf  rdf:resource= "#RouteNode"/>
</owl:Class>
<owl:Class  rdf:ID=SwitchEnd>
    <rdfs:subClassOf  rdf:resource= "#Node" />
    <rdfs:subClassOf  rdf:resource= "#RouteNode"/>
</owl:Class>
<owl:Class rdf:ID=LoopBegin>
    <rdfs:subClassOf  rdf:resource= "#Node" />
    <rdfs:subClassOf  rdf:resource= "#RouteNode"/>
</owl:Class>
<owl:Class rdf:ID=LoopEnd>
    <rdfs:subClassOf  rdf:resource= "#Node" />
    <rdfs:subClassOf  rdf:resource= "#RouteNode"/>
</owl:Class>
```

In OWL language, binary relation from one class to another class can be defined using owl:ObjectProperty. To specify process description, object property is defined as a transition to denote the connections between nodes.

```
<owl:ObjectProperty rdf:about= "transition" >
    <rdfs:domain rdf:resource= "#Node" />
    <rdfs:range rdf:resource= "#Node" />
</owl:ObjectProperty>
```

4.5. Context Model Elements

Context modeling elements include place, time, actor and other context information. We use "objectPropery" in OWL language to describe context.

Task execution time is defined as ExecuteAt with OWL as:

```
<owl:ObjectProperty  rdf:about= "ExecuteAt" >
    <rdfs:domain  rdf:resource= "#TaskNode" />
    <rdfs:range     rdf:resource= "#Time" />
</owl:ObjectProperty>
```

Note that TaskNode is the task node and Time is the time class.

Task execution place is defined as ExecuteIn in OWL as:

```
<owl:ObjectProperty  rdf:about= "ExecuteIn" >
    <rdfs:domain  rdf:resource= "#TaskNode" />
    <rdfs:range     rdf:resource= "#Location" />
</owl:ObjectProperty>
```

Note that TaskNode is the task node and Location is the geographical location class.

Relation between tasks and actors is defined as ExecueBy by OWL:

```
<owl:ObjectProperty  rdf:about= "ExecuteBy" >
    <rdfs:domain  rdf:resource= "#TaskNode" />
    <rdfs:range     rdf:resource= "#Agent" />
</owl:ObjectProperty>
```

Here Agent is the actor class.

Besides the above three typical contexts, users can define their own object property in accordance with OWL grammar to correlate context with object. User defined object property can greatly enhance the descriptiveness and scalability of our proposed modeling approach so as to make use of any context to model workflow.

We use constraint mechanisms and boolean combination of class in OWL to express context. In OWL, boolean combination mechanisms include 'complement', 'and', 'or' operations of class. In context-aware workflow models, the context information is used as the constraint condition of a node class and the node is described as a subclass with the above constraint conditions.

5. System Implementation

We adopt jBPM 4.3 (open-source software for building workflow application) [12] to realize the workflow engine. jBPM was developed by jboss community and was written in Java. In Perspective of software engineer, jBMP is suite of Java API which can define, create, manipulate and manage the workflow.

The context provisioning platform (CPP) is a key subsystem of context-aware workflow system and the core of context provisioning platform is inference machine which maps low-level context information to high-level context. Because we use OWL to model the context information, Jena (an open source Java framework for building Semantic Web and Linked Data applications) [13] can help us to develop the context provisioning platform. Jena was developed by HP Labs Semantic Web Programme and provides a suite of API for RDF and OWL processing and includes a rule-based inference engine. The context knowledge database and context model database can be deployed into MySQL DBMS. Context program application interface (Context API) is written in Java and by which workflow engine can get high level context.

Other components including workflow administration tool, workflow designer tool, workflow simulation engine and analysis platform are also written in Java. The applications that are called by workflow engine can be written in any program language and can be wrapped in Web service or Web API. The workflow client GUI is represented by JSP and Servlet and the Java EE container is Tomcat which is popular software for Java web development.

6. Modeling Example

We use simplified machine maintenance in an intelligent factory to illustrate our context-aware workflow model. In an intelligent factory, main parts of the machine to process the product will gradually wear out. When degree of wear grows over some threshold, the Tool needs to be replaced. In this scenario, location of the machine, wear degree of Tool, location of the people to replace the tool, location of Tool are context information. Location of these objects is tracked by RFID tags and Tool wear degree is monitored by sensors. When Tool is to be replaced, we firstly check whether there are spare Tool parts in the factory, and if there are spare parts, Tool will be replaced by the nearest operator according to context information such as machine location, location of operator, location of Tool etc. Or else, spare parts of Tool stock subflow are initiated and Tool will not be replaced until the spare parts are stocked. After Tool is replaced, the machine maintenance process instance is completed. With ontology tool Protégé [14], context-aware workflow model of the above process is illustrated in Fig. 4.

As can be seen from Fig. 5, there are 8 nodes in this workflow. "*start*" is the beginning node, "*Tool_expiry_detection*" is an automatic task node, "*Spares_check_SwitchBegin*" and "*Spares_check_SwitchEnd*" are switch node which form a switch selection structure together, "*Purchase_spares*" is a sub-process node, "*Replace_tool*" is a human task node, and "*end*" is the termination node.

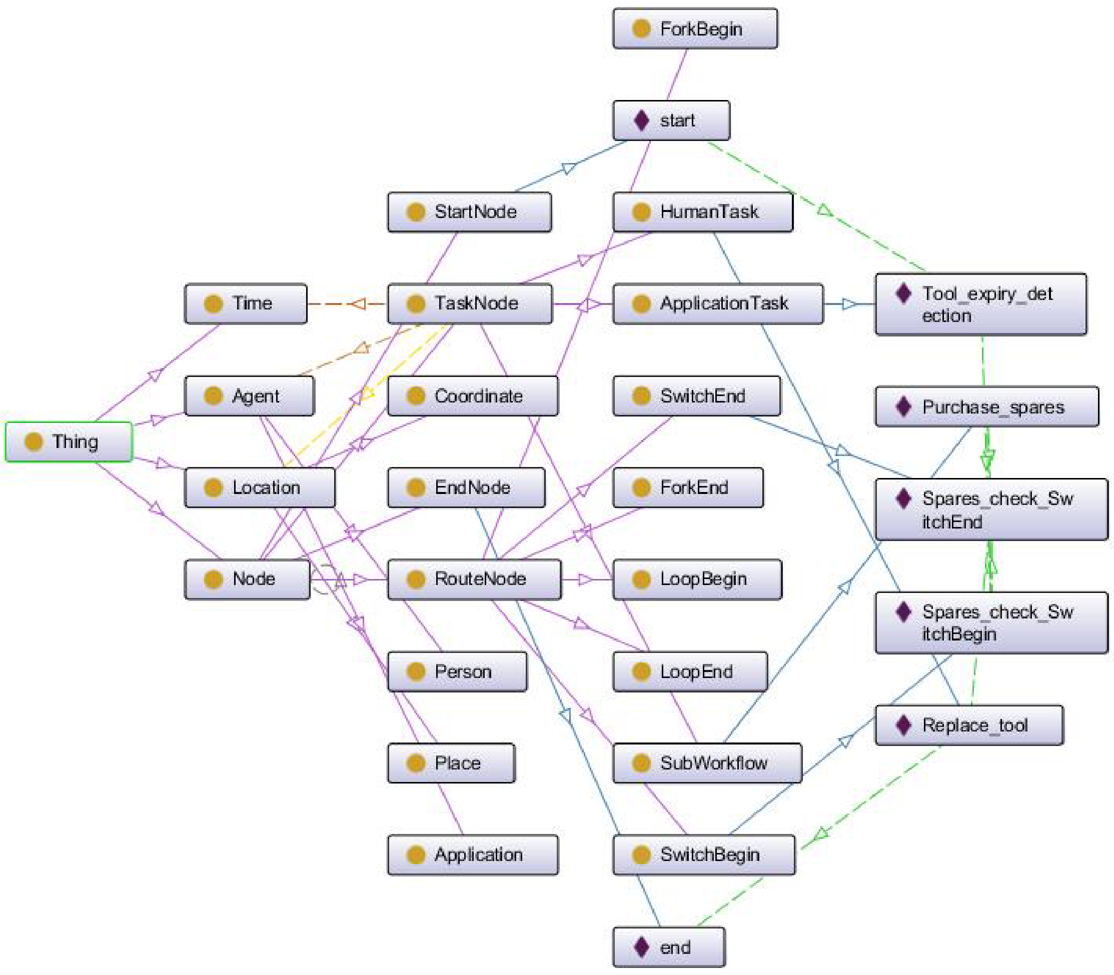


Fig. 4. Smart factory machine maintenance workflow modeling using Protégé.

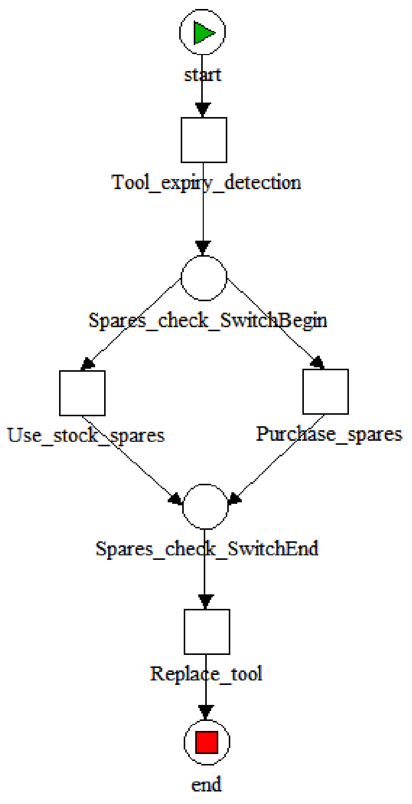


Fig. 5. Machine maintenance workflow diagram.

Execution of this process is as follows:

The process begins from the "*start*" node, then proceeds to the "*Tool_expiry_detection*" node which is an automatic task node to recurrently and frequently check wear degree of Tool: when the wear degree is below the threshold, the detection process is circled, when the wear degree is over the threshold process quits the detection process and reports the Tool status and moves forward to the following switch nodes.

"*Spares_check_SwitchBegin*" node does conditional selection and confirms spare parts location according to RFID tag data. If there are spare parts in the factory, the process goes to "*Spares_check_SwitchEnd*" node, otherwise goes to "*Purchase_spares*" sub-process. "*Spares_check_SwitchBegin*" node corresponds to context information utilization schema 1 that uses context information to select the switch. "*Purchase_spares*" sub-process corresponds to context information utilization schema 3 that initiates process instance with context information.

"*Spares_check_SwitchEnd*" node is to make the process structure better. When process reaches this node, the process should move to the following node, namely the "*Replace_tool*" node.

The "*Replace_tool*" node is a human task node. Operator first fetches the spare parts and then goes back to the machine to replace the old Tool with the spare part. In the factory, tasks need to be assigned to the operator according to the context information. Context-aware workflow engine assigns operator according to context information such as machine location, location of spare parts, operator status with intelligent optimization algorithms. The "*Replace_tool*" node corresponds to context information utilization schema 2 that assigns workflow tasks according context information.

After the "*Replace_tool*", the process move to the "*end*" node and the machine maintenance process is finished.

7. Conclusions

In the pervasive computing and mobile computing environment, business process and context information are closely linked, context-aware workflow management has become an important demand. Context-aware workflow technology makes intelligent management of business process possible in the ubiquitous computing and mobile computing environment. But it is lack of standard context-aware workflow reference model, go against context-aware workflow product standardization and interoperability.

For this reason, this paper proposes novel context-aware workflow management system framework. In the framework, CWfMS is different from the traditional workflow management system and it introduced into the context-aware middleware. Context-aware middleware shielding the various bottom sensors work details for the application and directly provide high-level context information, simplified the upper application development, make the workflow applications focus on using high-level context, so as to realize the intelligent workflow management.

In this paper, a context-aware workflow modeling language based on OWL language is proposed. This language uses the ontology of OWL language that can describe the complex context in the process. Modeling elements include process elements and context elements described with OWL class and its properties. This approach has the following advantages: 1) it can express complex contexts and context information can be explicitly expressed in the process model; 2) it has good language structure. To understand and model easily, parallel route, select route and loop route all need a starting node and an ending node in pairs. 3) it is platform independent. Because OWL language is an international standard modeling language, model sharing and interoperation can be easily established in different systems and software platforms.

The context-aware workflow management system can be applied in many fields, e.g. smart factory, smart medical care, smart agriculture and so on. Along with the ubiquitous computing, mobile Internet, wireless sensor networks technology popularization, context-aware workflow technology will achieve even greater development, and will meet more and more complex business process management requirements.

Acknowledgements

The authors are grateful to the anonymous referees for their valuable comments and suggestions to improve the presentation of this paper. This work was supported in part by a grant from the National Natural Science Foundation of China (No. 61211130359)

References

[1]. A. K. Dey, Understanding and using context, *Personal and Ubiquitous Computing*, Vol. 5, Issue 1, 2001, pp. 4-7.

[2]. Workflow Management Coalition, Terminology & Glossary, Technical Report, *The Workflow Management Coalition (WfMC)*, February 1999.

[3]. L. Ardissono, R. Furnari, A. Goy, and G. Petrone, et al, Context-aware workflow management, *Lecture Notes in Computer Science*, Vol. 4607, 2007, pp. 47-52.

[4]. L. Ardissono, R. Furnari, A. Goy, and G. Petrone, et al, A framework for the management of context-aware workflow systems, in *Proceedings of the 3rd International Conference on Web Information Systems and Technologies*, March 2007, pp. 80-87.

[5]. L. Ardissono, A. D. Leva, G. Petrone, and M. Segnan, et al, Adaptive medical workflow management for a context-dependent home healthcare assistance service, *Electronic Notes in Theoretical Computer Science*, Vol. 146, Issue 1, 2006, pp. 59-68.

[6]. M. Wieland, P. Kaczmarczyk, and D. Nicklas, Context integration for smart workflows, in *Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications*, March 2008, pp. 239-242.

[7]. Y. Cho, J. Choi, and J. Choi, A context-aware workflow system for a smart home, in *Proceedings of the 2nd International Conference on Convergent Information Technology*, November, 2007, pp. 95-100.

[8]. J. Han, Y. Cho and J. Choi, A workflow language based on structural context model for ubiquitous computing, in *Proceedings of the International Conference on Embedded and Ubiquitous Computing*, December 2005, pp. 879-889.

[9]. F. Tang, M. Guo, and M. Dong, et al, Towards context-aware workflow management for ubiquitous computing, in *Proceedings of the International Conference on Embedded Software and Systems*, January 2008, pp. 221-228.

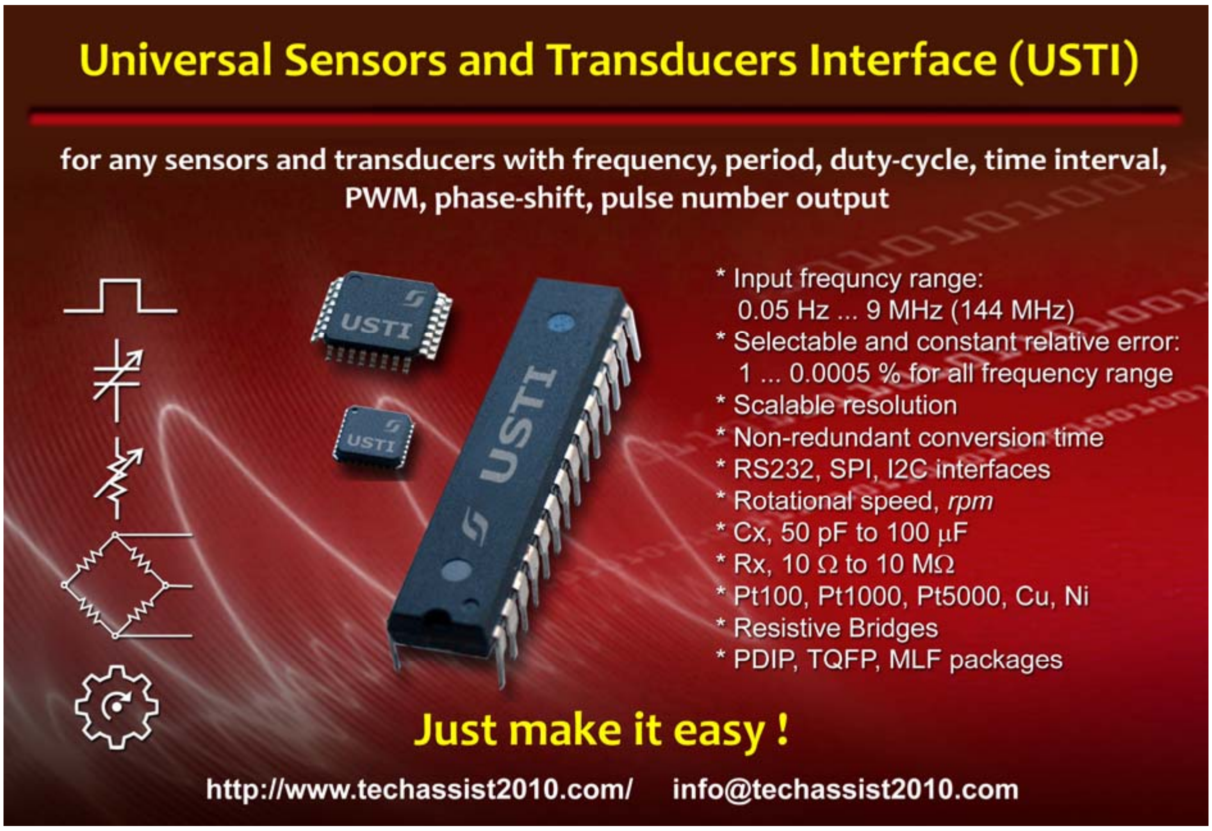
[10]. F. Tang, I. You, M. Guo, and S. Guo, Context-aware workflow management for intelligent navigation applications in pervasive environments, *Intelligent Automation and Soft Computing*, Vol. 16, Issue 4, 2010, pp. 605-619.

[11]. OWL Web Ontology Language Guide (http://www.w3.org/TR/2004/REC-owl-guide-20040210/).

[12]. jBPM Web Site (http://jbpm.jboss.org/).

[13]. Jena Toolkit Web Site (http://jena.apache.org/).

[14]. Protégé Web Site (http://protege.stanford.edu/).


Universal Sensors and Transducers Interface (USTI)
for any sensors and transducers with frequency, period, duty-cycle, time interval, PWM, phase-shift, pulse number output
* Input frequncy range: 0.05 Hz ... 9 MHz (144 MHz)
* Selectable and constant relative error: 1 ... 0.0005 % for all frequency range
* Scalable resolution
* Non-redundant conversion time
* RS232, SPI, I2C interfaces
* Rotational speed, rpm
* Cx, 50 pF to 100 μF
* Rx, 10 Ω to 10 MΩ
* Pt100, Pt1000, Pt5000, Cu, Ni
* Resistive Bridges
* PDIP, TQFP, MLF packages
Just make it easy !
http://www.techassist2010.com/ info@techassist2010.com