Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 211 (2008) 211–220
www.elsevier.com/locate/entcs

BPSL Modeler - Visual Notation Language for Intuitive Business Property Reasoning

Ke Xu [1]

Department of Automation, Tsinghua University
Beijing, China

Ying Liu[2]

IBM China Research Laboratory
Beijing, China

Cheng Wu

Department of Automation, Tsinghua University
Beijing, China

Abstract

The urgent need for reliable business applications demands the emergence of a powerful yet easy-to-use language for business property reasoning. The Business Property Specification Language (BPSL) and its supporting tool (BPSL modeler) are presented to address the issue. BPSL modeler facilitates the specification and understanding of business properties by simplifying the expression of complex logics and common behaviors in business processes and exploiting intuitive notations for property representation. It also serves as a key component of our method for model checking business processes. Important ideas and features of BPSL modeler are provided in this paper to help understand its effectiveness.

Keywords: Business Property Specification Language, Temporal Logic, Business Property Template, Formal Verification

1 Introduction

Driven by the growth of complexity in existing business systems and the urgent need for ensuring reliable business applications, it has been recognized to be a promising approach to integrate formal verification techniques like model checking [4] into business domains to help efficiently verify their business processes [13]. Specifying

[1] Email: xk02@mails.tsinghua.edu.cn
[2] Email: aliceliu@cn.ibm.com

doi:10.1016/j.entcs.2008.04.043

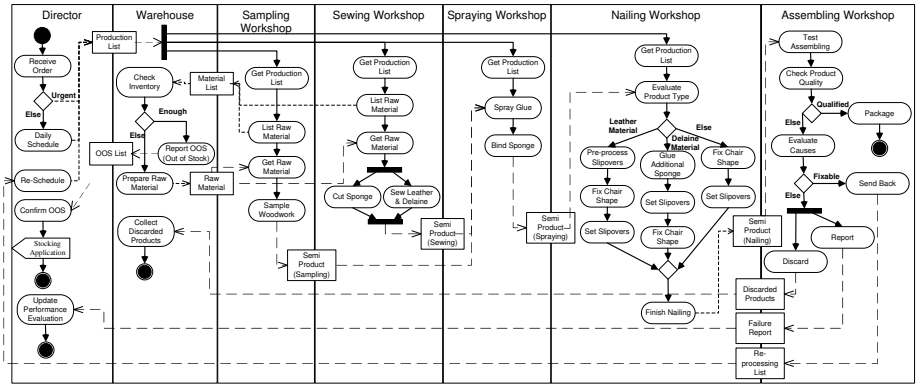

Fig. 1. Business Process for Sofa Manufacturing

ReportGeneration=G (¬reportOOS) → G((receiveorder ∧ urgent) → (F ((¬sentback U (sentback ∧ (X ¬sentback U (sentback ∧ report.finished)))) ∨ (¬sentback U (sentback ∧ (X ¬sentback U (sentback ∧ (X ¬sentback U (sentback ∧ report.finished)))))))) ∨ G (¬(¬sentback U (sentback ∧ (X ¬sentback U (sentback ∧ (X ¬sentback U sentback))))) ∧ F package.inexecution)))

Fig. 2. LTL Specification of ReportGeneration Property

user desired properties is a critical step in the application of model checking. These properties, as expressed in formulas such as temporal logics [4] of CTL or LTL, captures specific user requirements on business processes and are thus called business properties (in short, *BP*). However, the complexity and rigidness of logical formulas is a serious obstacle for business analysts to write and understand their desired properties since most of them are not logical experts. Therefore, tool support for intuitive and easy specification of business properties, automatic generation of their logical formulas are critical for the application of model checking business processes.

Take the following real scenario as an example. Figure 1 illustrates a partial model of a sofa manufacturing process in UML activity diagram. In the process, the factory director delegates manufacturing orders to different workshops. The assembly workshop assembles the semi-finished products and checks their qualities. In order to ensure the correctness of such a complex process design, an even more complex property may need to be specified with which the process model is supposed to comply. For example: "*Without considering the situation of Out Of Stock(OOS), whenever an urgent order is received a product must eventually be packaged if it is re-processed less than three times. However, a failure report must to be generated after its second sent back for re-precessing and before the product is re-precessed the third time*". This property can be captured with LTL as shown in figure 2.

The complexity of business properties in traditional temporal logics can be easily understood from the example. Therefore, the primary contribution of this work is the proposal of an easy-to-use visual notation language called BPSL and its supporting tool BPSL modeler that are tailored for the application in business domains. BPSL modeler enjoys the advantage of maintaining good usability and understandability for specifying business properties while still preserving their strict formal semantics to support the formal verification. These advantages are enabled by tailoring BPSL for specifying common behaviors in business processes and absorbing

existing business knowledge into the language. As a comparison, we will illustrate in section 3 how the above property can be more intuitively specified with BPSL modeler.

2 Related Works

An important working direction for facilitating temporal property specification is to provide visual extensions for existing logics [3][5][11]. This benefits users by helping understand different semantics of temporal operators with their visualized formalisms. On the other hand the Property Specification Language(PSL) [6], an IEEE standard in digital circuit community, focuses on reducing the complexity in property specification by providing a more flexible choice of temporal operators. When taking a close investigation in business domain, in REALM (Regulations Expressed as Logical Models) [7], an extension of propositional temporal logics is contained to specify compliance rules in business models. A domain specific model checking language tuned for business applications named Strix is proposed in [1]. The property specification in Strix directly uses CTL connectives to explore the business process model.

However, the above works still suffer from the following deficiencies: (1) When plural property is considered (e.g. the one in figure 2), providing visual notations alone is not enough for making a specification language easy-to-use and a property intuitive to understand; (2) Existing knowledge in specifying different business entities (e.g. *activity*, *resource*) and their relations (e.g. *response*, *exclusion*) are not fully exploited to facilitate business property specification and understanding; (3) It lacks the tool support for automatically generating formal semantics from intuitive business property representations so as to enable the quick integration between business process modelers and formal verification tools.

The visual notation language of BPSL is extended from PSL by specifically tuning it for business domains. Distinct features of BPSL modeler include: (1) BPSL provides an intuitive representation of business properties such that business people can easily understand them with their existing knowledge and experience; (2) By categorizing frequently used business property templates and providing the "push button" generation of their BPSL definitions, BPSL modeler absorbs existing business knowledge and facilitates business property specification; (3) BPSL modeler supports the auto-generation of underlying formal semantics of each property based on both the logic of CTL and LTL. Consequently, it not only ensures the preciseness of BPSL, but also facilitates the reasoning of business models with existing formal verification tools.

The rest of the paper is organized as follows. In section 2 the framework and basic concepts in BPSL modeler are introduced to help understand its ideas. In section 4, the features of BPSL modeler are explained in detail together with the analysis of corresponding characteristics in business property specification. Section 5 illustrates how BPSL modeler plays the role in our method of model checking business processes. Section 6 concludes the paper.

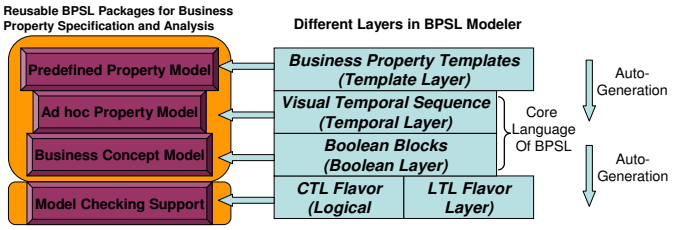

Fig. 3. Framework of BPSL Modeler

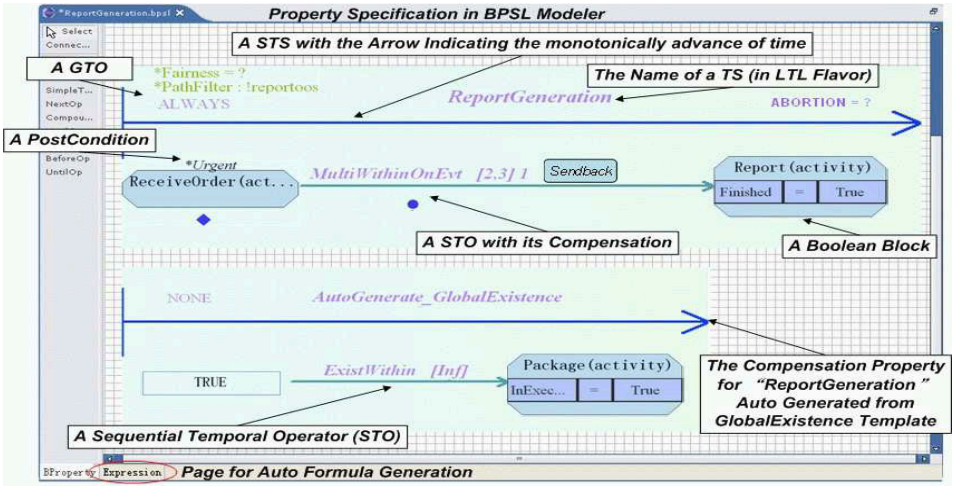

Fig. 4. Re-specify ReportGeneration property in BPSL Modeler

3 BPSL Modeler - Framework and Basic Concepts

To better understand the ideas in BPSL modeler for business property specification, figure 3 illustrates its framework. BPSL consists of a Boolean layer and a temporal layer. In Boolean Layer, Boolean Blocks (in short, *BB*) are basic elements for capturing the attributes of different business entities (e.g. activity, resource) and form a basic concept model for specific business domain. In temporal layer, Temporal Sequence (in short, *TS*) is the visual representation that specifies the logical relations between different business entities and the temporal constraints on specific business models. Above the two layers, BPSL modeler concludes frequently used business properties or patterns from business experiences in the form of Business Property Templates. BPSL modeler supports the push button generation of the BPSL definition for each template and in turn the CTL/LTL definition for each *BP* as its formal semantics. Consequently, a benefit of the framework is that property specifications in BPSL modeler can be easily reorganized to form reusable BPSL packages for property generation in different business application domains. Besides, the CTL/LTL formal foundation also facilitates the integration of business property specification with existing formal verification techniques since the reasoning of the two temporal logics have a wide tool support such as RuleBase [2], etc. A full syntax, semantics and visual notations of BPSL can be found in [14].

To better understand BPSL, figure 4 illustrates the BPSL implementation for the previous "*ReportGeneration*" property. The auto-generated formal semantics of this specification with BPSL modeler coincides exactly with the one in figure 2. We will explain the details of the specification in the incoming two sections by investigating the basic concepts and advanced features of BPSL modeler involved in this example.

- ***Boolean Block(BB):*** Represented by an octagon, a *BB* is a three-tuple consists of its name (e.g. *Report*), stereotype (e.g. *Activity*) and a set of atomic attributes (e.g. whether it is finished). It can thus be used to identify specific business entities that are qualified by the pre-defined conditions in *BB*. For example, the above *BB* of *Report* indicates a *Report* activity that has already been finished.
- ***Simple Temporal Sequence (STS):*** A *STS* specifies the temporal relation among a sequence of *BB*s or business properties along paths were time advances monotonically. As can be found in [14], BPSL supports 14 stereotypes of Sequential Temporal Operators (*STO*s) with different semantics to specify these relations in different situations. While some of the *STO*s have a direct mapping to basic temporal operators (e.g. *Next 1* for X, *AllWithin Infinity* for G, *PossiblyLeadsto BeforeInfinity* for EF, etc), others can be used to express rather complex temporal relations in a simple and compact manner. For example, the property in figure 4 is itself a *STS*. The *STO* of *MultiWithin OnEvt* specifies that when an *Urgent* order is *Received*, a *Report* should be finished for once between the second and the third occurrence of event *Send Back*.
- ***Compound Temporal Sequence (CTS):*** Different from *STS*, a *CTS* specifies the different logical relations (e.g. *And, Or, Not, Imply, IFF*) and predefined temporal relations (e.g. *Before* for the weak Until operator W [10], *After* for F, *Until* for U) between two *STS*s with corresponding Compound Temporal Operators (*CTO*).
- ***LTL and CTL Flavor:*** In order to enhance the expressiveness of BPSL and obtain a wider tool support for formal verification, Temporal Sequence (*TS*s) in BPSL can be interpreted in either LTL or CTL flavor. *TS*s in different flavors possess different available temporal operators in BPSL in order to avoid the confusion of their semantics. For example, the temporal sequence of the above *ReportGeneration* is a LTL flavored specification. As can be found in [14], different *STO*s are associated with different notations so as to distinguish their semantics in the context of CTL and LTL (e.g. *Within, MultiWithinOnEvent*, etc in LTL flavor and *CertainlyLeadsTo, PossiblyLeadsToBeforeEvent*, etc in CTL flavor). It is then *BPSL* modeler's job to automatically generate the logical formalisms for each property according to its semantics in different flavors and hide the complexity from users.
- ***Global Temporal Operator (GTO):*** Four types of *GTO*s are supported: *"(Possible) Always"* for G and *AG(EG)*, *"(Possible) Eventually"* for F and *AF(EF)*, *"Repeat"* and *"Never"*. *"Repeat"* and *"Never"* guarantees that the *TS* must hold at least n times or never holds in the business process.

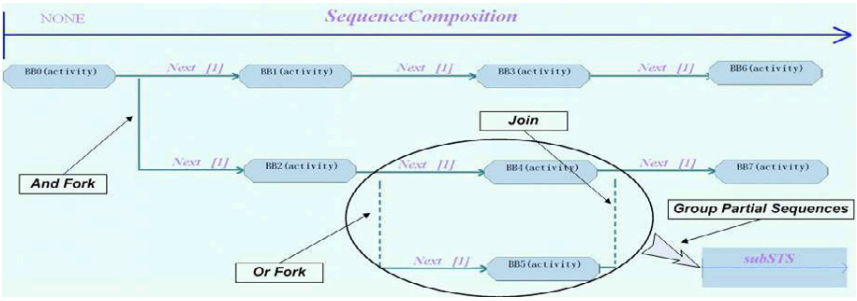

Fig. 5. Temporal Sequence Composition and Grouping

- ***Abortion:*** The abortion condition in a *TS* indicates the circumstance in which the evaluation of a *TS* should be forced to stop (e.g. by an external cancel event). In the above case, the question mark indicates that no abortion condition is explicitly specified.
- ***Postcondition:*** A postcondition can be associated with *BB*s in *TS*. It specifies whether the rest of the *TS* after a *BB* is necessary to be further evaluated. For example, the postcondition "*urgent*" associated with "*ReceiveOrder*" in the example indicates that only when the received order is urgent should the occurrence of report activity be evaluated.

4 Features of BPSL and BPSL Modeler

This section introduces the advanced features in BPSL modeler in accordance with the analysis of some characteristics in business property specification.

4.1 Sequence Composition and Grouping

Business people may not be familiar with logical reasoning, but most of them are familiar with popular process modeling techniques like UML Activity Diagram. Therefore, in order to make property visualization in BPSL more acceptable to business people by exploiting their existing experience, BPSL properties are also presented in a "process" oriented form. That is, Temporal Sequence (*TS*s) in BPSL not only specifies the temporal relations between two *BB*s or *BP*s, but can actually be composed to form a long chain of property sequence with And-Fork, Or-Fork and Join relations. Figure 5 illustrates such an example. The fork, join operators specify the different and/or relations when evaluating each branch in the *TS*. Besides, in BPSL the grouping of a partial *TS* is also supported which corresponds to the concept of sub-process in business process model. In figure 5 the details "*subSTS*" are hidden by grouping the corresponding parts in the original *STS*. Sequence composition and grouping enables the abbreviation of complex business property specifications in BPSL and flexibly adjusting the granularity of property representation.

4.2 Property Compensation

Traditional logical operators often have their relaxed versions to answer the question of "whether a property is still satisfied if certain condition does not hold". For example, *"P W Q"* is the relaxation of *"P U Q"* in that the weak until operator W does not order that Q must occur in the model while U does. In BPSL, such conditions like Q, which decide whether a property is satisfied strongly or weakly, are called Compensation Conditions (CCs). BPSL further generates the idea of property relaxation by enabling the flexible association of Compensation Properties (CPs) with the CC that each STO and CTO in BPSL may have. More specifically, a process model PM satisfies a business property BP with *(CC, CP)* iff $PM| = BP$ or $PM| = G(\neg CC) \wedge CP$. As a result, a strongly/weakly held property BP is a special case for property compensation in BPSL when its CP is *False*(as denoted by a rectangle)/*True*(as denoted by a diamond). For example, in figure 4 the CC for *Multi-Within OnEvt* operator is that there must be a third occurrence of event *Send Back* in the process. Consequently, the property of "$AutoGenerate_{G}lobalExistence$" which serves as its CP (as denoted by a circle) further specifies the rest of the semantics for *ReportGeneration* property that in case there is never a third occurrence of *Send Back*, a *Package* activity will be eventually executed. Compensation mechanism in BPSL effectively allows business analysts to specify their requirements by connecting different properties in their familiar *"if..then..else"* fashion. A reference of predefined CCs with temporal operators in BPSL can be found in [14].

4.3 Filtering and Fairness

Business processes can be rather complex so as to provide a full view of the daily businesses in an enterprise. Therefore when reasoning a business process it is often desired that redundant information (e.g. exception handling) which may not be the primary concern for business analysts can be neglected to avoid reaching wrong conclusions. Thus it will be of great help if the granularity of business reasoning can be flexibly controlled in the step of business property specification. BPSL provides this mechanism by supporting filter and fairness conditions. Borrowed from model checking [4], the fairness condition specifies that the evaluation of business property will only be done on process execution paths where it can hold infinitely often. On the contrary, the filter condition specifies that all the execution paths in the business process along which the filter condition may be satisfied will be neglected in the property evaluation. For example, while the example in figure 4 does not contain any fairness condition, its filter condition implies that the *ReportGeneration* property will not be evaluated in unusual cases where *Out Of Stock* may happen.

4.4 Business Property Templates

Years of practices and researches in business domain form a considerable accumulation of relevant experiences. BPSL modeler provides the capability of capturing these existing knowledge in the form of business property templates to facilitate both property specification and understanding. Four pre-defined common categories of

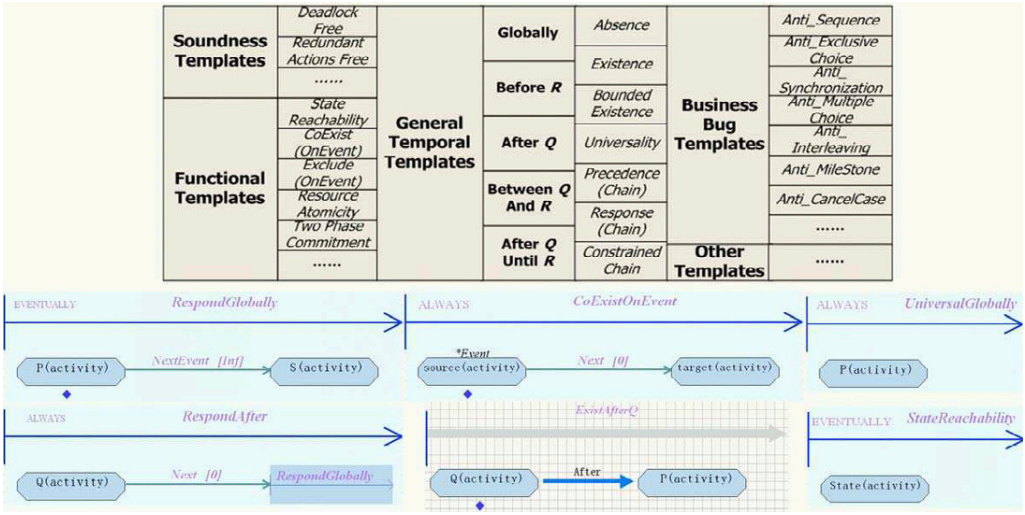

Fig. 6. Business Property Templates and Implementation Example

business property templates (Soundness Templates, General Temporal Templates, Business Bug Templates and Functional Templates) are concluded in BPSL modeler based on which frequently used business patterns can be automatically generated and reused. The definition of these business property templates can be automatically implemented within the expressiveness of BPSL and can in turn be formally interpreted in CTL/LTL with BPSL modeler. With the size limitation, figure 6 illustrates several examples of these templates and their implementations.

Soundness template concludes the common workflow soundness [8] definitions that each business process may satisfy; General temporal template implements the general temporal patterns based on the survey result in [9] and their semantic mapping [10] on CTL/LTL; Business bug template corresponds to traditional workflow patterns [12]. Each bug template is used to falsify a true realization of a specific workflow pattern in the business process model; Functional template captures some useful functional requirements in business processes. For example, *ResourceAtomicity* can be used to impose constraints on the process such that "for certain kind of resource (like money), it should never be created or destroyed in the process".

5 BPSL Modeler as a Component for Model Checking

BPSL Modeler is a key component of our OPAL (Open Process AnaLyzer) toolkit for model checking business processes (figure 7). In OPAL, business processes in different modeling techniques are formalized with Milner's Pi Calculus through standard Formalizer Interfaces. On the other hand, a set of GUI Interfaces are provided to connect property specification in BPSL modeler with process modelers. For example, in our current implementation, the GUI/Formalizer interface for Websphere Business Integrator (WBI) is provided such that not only the semantics of business processes modeled with WBI can be automatically formalized, but also Boolean Blocks in BPSL can be directly generated by drag-and-dropping different WBI

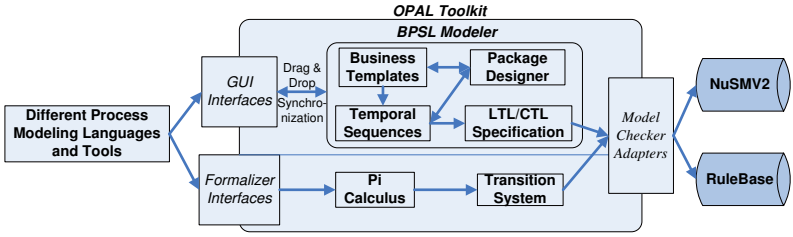

Fig. 7. BPSL Modeler as a Component for OPAL Toolkit

elements into BPSL modeler. These Boolean Blocks are then synchronized with corresponding elements in WBI so that they are always referencing to the same business entities. The relations between Boolean Blocks can either be generated from templates, or built directly with Temporal Sequences. In BPSL Modeler, a Package Designer is further implemented to store and edit user customized business properties and templates as a knowledge base for their later reuse. Available model checkers can thus be integrated with OPAL by transforming he auto-deduced transition system for process models and the LTL/CTL formulas into the format they accept through Model Checker Adapters.

6 Summary

Business property specification is a major step in reasoning business process models and ensuring its reliability. In this paper, the visual notation language of BPSL and its supporting tool BPSL modeler are proposed to enable the intuitive specification of business properties and facilitate the reasoning of business process models. BPSL modeler simplifies the complexity of business property specification by taking different business characteristics into consideration, e.g. the visual representation, property compensation and filtering, etc. It exploits existing knowledge in the practices and researches in business domain to make property specification of minimum efforts. In BPSL modeler, both the logics of LTL and CTL are supported and formal semantics can be auto-generated for each business property template and business property so as to ease the integration between BPSL modeler and existing formal verification tools. Our future work will include extending the application of BPSL modeler into more real cases to enlarge its values.

References

[1] Bard, B. *Seeing by Owl-Light - Symbolic Model Checking of Business Requirements*, Technical Report, IBM T.J. Watson Research Lab, 2002.

[2] Beer, I.S., B. David, et al. *RuleBase: Model checking at IBM*, International Conference on Computer Aided Verification, 1997, 480-483.

[3] Brambilla, M., A. Deutsch, et al. *The role of visual tools in a Web application design and verification framework: A visual notation for LTL formulae*, Lecture Notes in Computer Science, **3579**, 2005, 557-568.

[4] Clarke, E.M., Jr. O G., D. A. Peled, *Model Checking*, MIT Press, 1999.

[5] Del, B., A. L. Rella, E. Vicario, *Visual specification of branching time temporal logic*, Proc. 11th IEEE Symp. on Visual Languages, 1995, 61-68.

[6] Geist, D., *The PSL/Sugar specification language a language for all seasons*, Lecture Notes in Computer Science, **2800**, 2003, 3.

[7] Giblin, C., Y. Liu, et al. *Regulations Expressed As Logical Models (REALM)*, 18th Annual Conference on Legal Knowledge and Information Systems, IOS Press, Accepted, 2005.

[8] Hofstede, A., M. Orlowska, J, Rajapaske. *Verification problems in conceptual workflow specification*, Data and Knowledge Engineering, **24(3)**, 1998, 239-256.

[9] Matthew, B.D., S. A. George, C. C. James. *Patterns in Property Specifications for Finite-State Verification*, Proc. 21st International Conference on Software Engineering, 1999.

[10] Property Patterns, 2005, http://patterns.projects.cis.ksu.edu

[11] Rao, A. C., A. Cau, H. Zedan, *Visualization of Interval Temporal Logic*, Proc. 5th Joint Conference on Information Sciences, 2000, 687-690.

[12] van der Aalst, W.M.P, ter Hofstede, A.H.M, et al. *Workflow patterns*, Distributed and Parallel Databases, **14**, 2003, 5-51.

[13] Wang, W. L., Z. Hidvegi, et al. *E-process design and assurance using model checking*, Computer, **33(10)**, 2004, 48-53.

[14] Xu, K, L. Ying, W. Cheng. *A Property Specification Language for Verifying Business Process Models*, IBM Technical Report, RC23830(C0512-005), 2005.