Ontology Reasoning and Services Composition Verification towards O-RGPS Requirement Meta-Model

Zhenxing Xu[1]
[1]Chengdu Institute of Computer Applications
Chinese Academy of Sciences
Chengdu, China
freemanxzx@163.com

Jinzhao Wu[2,3]
[2]Beijing Jiaotong University, Beijing, China
[3]Guangxi University for Nationalities, Nanning, China
himrwujz@yahoo.com.cn

***Abstract*—the correctness and reliability of requirement specifications are vital to cost and success of developing a software system, especially for networked software within complex and dynamic change environment. In this work, the solution to verification of O-RGPS, which gives a meta-model for requirement modelling of large-scale networked software, is proposed. Firstly, adopting hybrid reasoning way, consistency of domain knowledge is checked and then inference is made to acquire potential information. Moreover, web services aggregation in S-layer is described with BPEL4WS, which is converted to promela presentation, and then reliability problems such as deadlock, reachability and so on are verified with model checker tool SPIN. As a result, it makes sure that the final requirement specifications of O-RGPS framework are faithful.**

Keywords-O-RGPS; ontology reasoning; model checking; web services; requirement verification

I. INTRODUCTION

Requirements phase plays a very important role on software engineering and is the key point for the quality of software project. With software system networking and more and more complex, requirements elicitation and analysis also become more and more difficult. The traditional requirements modelling methods such as KAOS[1], FeauRSB[2], ODE[3] and so on cannot be well adaptable to the conditions of networked requirements engineering.

A unified requirements meta-model frame O-RPGS has been proposed in [4, 5], providing a requirements modelling guidance for large-scale networked software system. Based on domain knowledge, it makes layered modeling and dynamic evolution from the view of role, goal, process and service and finally gets the functional and nonfunctional (including context and trustworthy requirements) requirement specifications on networked software. O-RGPS models the common requirements of a family of systems, and gives feasible modelling methods for these requirements by aggregating web services. Nevertheless, in this frame, verification mechanism has not been provided and the correctness of the meta-model cannot be ensured. Whether or not requirements specification is valid is of crucial importance to a certain model method. For this reason, applying ontology technologies and model checking to the verification of this meta-model frame will be discussed in this paper.

The structure of the article is organized as follows. Section 2 makes a brief introduction to O-RGPS. Section 3 discusses description logic-based and rule-based ontology reasoning and then gets a hybrid reasoning way. Section 4 constructs a message mode and gives a verification solution to S-layer. Technology architecture is drawn in Section 5 and, finally, the conclusions and our intentions to future work are exposed.

II. BACKGROUND

In this section, Briefintroduction to layered model of O-RPGS is given. Every alphabet stands for a layer model respectively. The letter O denoting domain knowledge means its domain-dependent modelling mechanism. O-layer is the basis of the whole framework and furthermore goes through other layers. It provides terminologies and basic roles for domain data sharing and interoperability. Subsequently from Role layer to Service layer, requirements specification is refined gradually at layer level. The Role layer, abbreviated for R-layer, characterizes the organization, roles and actors in problem domain, and describes the interaction and cooperation among them. The system goals, such as role's goals and actor's personal goals, are distilled and decomposed in the Goal layer. Meanwhile, some constraints and their relation are built up. After G-layer modelling, functional goals and nonfunctional goals will be acquired. The Process layer, written as P-layer in short, distinguishes atomic processes and composite processes and defines the input/output together with precondition/effect of the processes, achieving goals. The Service layer, namely S-layer, aggregates related web services through network to realize the result of P-layer.

III. ONTOLOGY REASONING

Since domain knowledge is the data source of other layers, so whether there exists inconsistent information is vital to following requirements modelling. For example, in an urban public transport system, much station information should be contained in the knowledge base. Provided that a terminology A(a station name) exists in initial knowledge base. But afterwards the station is disabled due to urban redevelopment. Then the negation of A will be added to the

978-1-4244-6542-2/$26.00© 2010 IEEE

knowledge base. The contradictory terminologies lead to opposite consequences.

A. O-layer Representaion

In order to check consistency of O-layer, domain knowledge should be well represented with computer-processable description language. Web Ontology Language (OWL)[6] recommended by W3C is designed for applications that need to process the content of information instead of just presenting information to humans. Based on description logic, its sublanguage OWL DL not only has good expressivity for domain knowledge, but powerful reasoning competence for checking consistency. Therefore, OWL DL is adopted to be responsible for representing domain knowledge of O-lay in this paper.

B. Description logic-based Reasoning

Description logic [7] is a decidable subset of first order logic and unifies some typical representation methods and semantic data models, including frame-based, semantic web and object-oriented thinking. In description logic, the knowledge base is divided into two parts: Tbox and Abox. The former stores terminologies and the latter contains instances and roles. With different constructors set, different expressivity and reasoning ability are possessed by different types of description logic. The most fundamental type is ALC from which others are all obtained. Using OWL DL, such abilities of SHOIN(D) can be applied to ontology reasoning, supporting transitive and inverse roles, role hierarchies, nominal concepts, unqualified number restriction and data type. The main reasoning tasks below can be done.

- Checking consistency of Tbox, i.e., determining if there exist contradictory concepts. For the above knowledge base of urban public transport system, the inconsistent concept A and the negation will be checked and excluded.
- Judging which concept a certain instance belongs to. This function can be used to checking consistency of Abox.
- Inference. If Tbox and Abox are both consistent, new pieces of domain knowledge can be inferred on their basis. But if inconsistent information is not eliminated, it will lead to logic explosion, i.e. any information can be inferred. This makes no sense.

The first two take responsibility of checking inconsistency of domain knowledge and the last one mines potential information. Therefore, the way of description logic-based reasoning makes sure the correctness of domain knowledge of O-layer.

C. Rule-based Reasoning

Other than the correctness problem, inference task may be also important for O-layer due to much potential information useful for modelling. Although description logic-based means seems to do this, it can do nothing to the following instance.

Assume that some pieces of informal information listed below have been contained in knowledge base of urban public transport system.

- Temperature (a).
- MinCriticalTemp (b).
- IsLowerThan (a, b).
- YellowAlarm(c).
- Annunciator (d).
- yeildYellowAlarm(d, c)

The symbols a, b and c denote instances of related concepts. Now the annunciator should yield yellow weather warning when temperature is lower than minimum critical value, which is described semiformally as follows:

Temperature(a)∧minCriticalTemp(b)∧isLowerThan(a, b)∧YellowAlarm(c)∧Annunciator(d)

→ yeildYellowAlarm(d,c).

Such a produce rule cannot be afforded in Description logic system. Hence, a new rule-based reasoning mechanism should be injected into inference. Considering the owl representation of O-layer and Semantic Web Rule Language (written as SWRL in short)[8] based on OWL Lite and OWL DL, rule-based reasoning is easily applied to owl ontologies. In SWRL, body is antecedent and head is consequent. When body is satisfied, then head will be obtained. Atom is basic constitute element of body and head, and has four forms as follows:

- C(x). C is a class (or concept) description and x is a variable.
- P(x, y). P means properties of owl while x and y can be replaced with variables, instances and data value.
- SameAs(x, y) denotes x equivalent to y.
- DifferentFrom(x, y) has opposite meaning with SameAs(x, y).

The weather warning rule as previously mentioned can be described formally with SWRL.

Temperature(?a)∧minCriticalTemp(?b)∧isLowerThan(?a,?b)∧YellowAlarm(?c)∧Annunciator(?d)

→ yeildYellowAlarm (?d, ?c).

D. Hybrid Reasoning

Synthesizing virtues of the two methods above, a hybrid reasoning way is proposed in this section. The reasoning steps have been summarized briefly below:

- After loading ontologies, an option that ontoglogies is preprocessed can be selected. If "no" is selected, go to step four directly, otherwise continue the next step.
- Reasoners and reasoning tasks are selected. Reasoners include some open source reasoners such as pellet, FaCT++, and reasoning tasks contain consistency checking, inference and so on.
- Determine whether ontologies are updated.
- The following steps are the same as standalone rule-based reasoning way.

Hybrid Reasoning integrates loosely description logic reasoning and rule-based reasoning. They both not only work together, but do it on its own.

IV. S-LAYER VERIFICATION

The semantic web services, deployed in networks and published in a service repository by their service suppliers,

are described and aggregated in S-layer to realize the P-layer, which can be viewed as the solutions for common domain requirements. Compositing flexibly registered web services as requirements changing is the feature of networked software. With the software reuse technology, development cycle can be reduced greatly enormously. Nevertheless, some problems are also brought to the composition of services.

- Deadlock. Web services which may be invoked simultaneously by others become the critical resource, competition for which may cause deadlock. For instance, the service for operating the count of tickets in the train ticketing system is the critical resource for tickets terminal service, which easily leads to the occurrence of deadlock.
- Livelock. Since services are distributed on the Internet, every one is in the dynamic environment. In contrary to deadlock, the states of web services competing for critical resources change continuously dynamically.
- Safety. It indicates the properties which do not conflict with web services aggregation.
- Liveness. The system must satisfy some properties, which usually indicates the function or performance needs.
- Reachability. It denotes which states and paths can be reached

The verification to the above problems of S-layer can be done utilizing model checking [9]. After transforming the description of web services composition to the input of a certain model checker, the properties expressed with temporal logic can be checked by the model checker. The conversion solution and a verification frame are proposed for the rest of this paper.

A. *The description of S-layer*

Business Process Execution Language for Web Services (BPEL4WS or BPEL) is a standard executable language for specifying interactions with web services and can be used to composite and orchestrate web services into a new web service. As the description language for S-layer here, it has basic modeling elements such as variables and its types, basic activities, control structures which elaborates the interactive behaviors. The concrete syntax and use manual can be found in [10].

B. *SPIN*

SPIN[11] is one of the most popular and powerful model checkers. It is designed for the verification problem of large-scale concurrent software system. The input language proemla has detail language ingredients like C programming language to model system actions and states, and linear temporal logic (short for LTL) has the function that expressing system properties. For the reason of the effective verification algorithm and open source thinking, it is adopted to afford the verification task towards the reliability of S-layer. Therefore, Transformation between BPEL4WS and promela description should be made and then analyzed below.

C. *From BPEL to Promela*

The transformation is separated into two sections. The conversion for variables and datatypes is included in the first section and activities are transformed in the second part. 1) & 2) corresponds to the former and then 3) & D to the latter.

*1) Avoidance of naming confiction:*owing to different reserved keywords in two description languages, naming conflicts may occur potentially. For example, "mtype" is a keyword and declares a message type in promla but not in BPEL, the transfromation without no deal between them will lead to invalid syntax problem. A form of "BPEL_validname_type" is devised here. The prefix "BPEL" indicates the source file and the postfix denotes the source type of the name, which has several types like variables type and all the activies types and so on. Each type is identified uniquely. Towards above instance, "mtype" can be renamed to "BPEL_mytpe_var" if "mtype" is considered as a variable in BPEL.

2) Conversion of variable types: Three types are defined in BPEL: wsdl message type that defines communication message format among web services, xml schema simple type and xml schema element. The common types such as boolean, int, short, byte, unsigned int between BPEL and promela can be mapped directly. Other built-in types possessed by xml schema simple type like string, dateTime, time, date are mapped to the type mtype. The exception that no float type is contained in promela should be considered depending on its occurrence context.

- As a judgment value of a certain property. In such "bpws: getVariableData ("var","part")>23.6" as the judgment condition of the attribute "transtionCondition" in the tag <source>, the float value will be discretized with two integral value, one of which is less than original float value and the other larger. This way does not affect the function of condition expression. On the contrary, it stems from and reflects the meaning of judgment function.
- As a right value in the assignment activity. In this circumstance, it is converted to the type mtype.

For message type and xml schema element, if there is no concrete assignment operation, they are also mapped to the type mtype. Otherwise, the keyword typedef is used to define the elements of the two.

3) Message Mode

Besides, the activities including basic activities and composite activities also need to be converted. Here a message mode is constructed, which is the core of activities transformation.

Definition 1. A message node is a triple, MN=<C_{in}, Act, C_{out}>.

- C_{in} is the set of in-message channels of a certain activity unit.
- Act denotes a certain activity unit.
- C_{out} is the set of out-message channels of a certain activity unit.

There are two sources for message channel, one is the sequence stream and the other is the link. The former is a message type that is used to reduce sequence activities to

concurrent activities. The latter is the one that links a source activity and a target activity.

Definition 2. A message stream is an orderly sequence of message nodes, which interconnects together through message channels. It can be written as the form as MS = $MN_1MN_2...MN_n$.

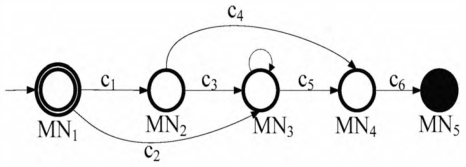

Figure 1. Example of a Message Stream

As is demonstrated in the figure 1, there is a message stream MS = $MN_1MN_2MN_3MN_4MN_5$, which consists of five message nodes through outer message channels token with solid arrows. A dotted arrow indicates the inner message of activities such as "while" activity. A circle stands for a message node, in particular a bicyclo for a starting node and a black circle for an end node. The five nodes can be formally written as MN_1=< {c_0}, Act_1, {c_1}>, MN_2=< {c_1}, Act_2, {c_3, c_4}>, MN_3=< {c_2, c_3}, Act_3, {c_5}>, MN_4=< {c_4,c_5}, Act_4, {c_6}>, MN_5=< {c_6}, Act_5, Ø>.Note that it is impossible to have loop circuits in the graph because a source (or target) activity is forbidden to become a target (or source) activity in a link of BPEL.

D. Conversion of Activities

Reclassification for activities in BPEL is to be made here for simplicity. Three activities are partitioned according to transformation rules, namely P-activity, S-activity and C-activity and summarized as follows.

- P-activities. receive, reply, invoke, switch, while, pick, throw.
- S-activities. assign, empty, terminate, wait.
- C-activities. sequence, flow.

Every P-activity is converted to a standalone process of promela and the one belonging to S-activity to a sequence of statements and then conversion of C-activities is melted into them. What needs to be pointed out is that P-activity conversion rule described above is just abode when it is not embedded inside any other activity; otherwise, it is the same as S-activity.

*1) Conversion of P-activities:*The specific conversion way is shown in the table Ⅰ and a activity correspondes to a process of premla.

TABLE I. P-ACTIVITIES CONVERSION

Activity type	Conversion code
receive	proctype BPEL_receive_* {send_stats;}
relpy	proctype BPEL_reply_* {rec_stats;}
invoke	proctype BPEL_receive_* {rec_stats; others_stats;}
switch	proctype BPEL_switch_* {do ::(condition 1);break; :: … :: (condition n);break; od}
while	proctype BPEL_ while_* {do R(P)/Stat(S); (!condition)->break; od}
pick	proctype BPEL_pick_* {if ::(condition 1)->R(P1)/Stat(S1); … ::(condition n)->R(Pn)/stat(Sn) fi}

throw	proctype BPEL_throw_*(mtype error)
	{
	printf("%e occurs",error);
	}

The wildcard character "*" delegates the integral number counting from zero. The "send_stats" & "rec_stats" & "R(P)" mean separately a statement sequence of sending, receiving a message and nested activity, and then "other_stats" & "Stat(S)" both stand for codes of the behavior statements of the activity itself.

*2) Conversion of S-activities:*According to the synmantics of S-activity, every one is converted to statements.The activity "assign" corresponds to a assignement statement and "empty" to "skip", "terminate" to "end" and then "wait" to a "while" constrol structure with a false condition.

*3) Conversion of C-activites:*Using the message mode and rendezvous communication mechanism, the sequential structure can be reduced to the concurrent structure. Assume that Act_1 and Act_2 are contained in the tage <sequence> without any message communication between them. Then a rendezvous port should be constructed to make a rendezvous communication. The conversion pseudocode is presented in table Ⅱ.

TABLE II. REDUCTION FOR SEQUENCE STRUCTURE

Activity type	Conversion code
<sequence> Act1 Act2 </sequence>	chan BPEL_Act1_to_Act2_Syn = [0] of {mtype }; proctype BPEL_Act1_* {send_stats;others;} Proctype BPEL_Act2_* {rec_stats;others;}

In concurrent structure, every activity is converted to a standalone process. BPEL_Act1_to_Act2_Syn namely is the rendezvous port, whose capacity is zero. A send statement and a receive statement have been positioned respectively in Act_1 and Act_2 to form a synchronization relation, which then can run concurrently. In the tag <flow>, a tag <link> to link a pair of activities corresponds to a message channel, whose capacity is determined by out-message streams and in-message streams.

V. RELATED IMPLEMENTATION TECHNOLOGIES

Many open source APIs and tools exists and can be utilized for implementation. On the one hand, OWL API and Protégé-OWL API both provide well-defined interfaces for operating ontology files or database and integrate typical

open source reasoners such as pellet and FaCT++. On the other hand, Jess is considered as the back-end rule-based reasoning engine and its input format of ontologies is incompatible with the common formats, and therefore the transformation between two formats should be done. This realization is furnished by Protégé-OWL API. Besides, SWRL-Bridge takes responsibility for format conversion of swrl rules ontologies. In such a way, the technology roadmap can be drawn and presented in figure 2.

As is illustrated in the graph, three layers architecture is designed apart from use interface. The data layer takes charge of data storage, data serialization and deserialization. All conversion tasks such as owl and swrl ontologies format, BPEL2Promela are accomplished in the conversion layer. Reasoning&Verification layer is responsible for hybrid reasoning and web services composition verification. Note that the DIG module of hybrid reasoning gives a chance that it visits remote reasoners via network.

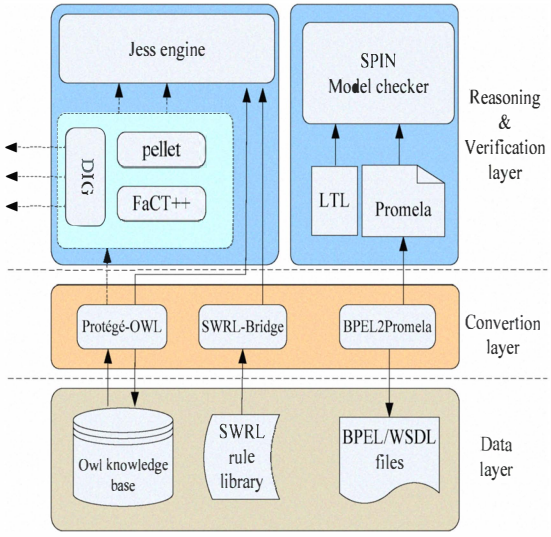

Figure 2. A Techonology Roadmap

VI. Conclusions and Future Work

In this paper we proposed a verification framework for O-RGPS to make sure that consistent domain knowledge base can be acquired and the result of S-layer of the model is reliable. The transformation based on the message mode from BPEL to promela was also constructed. Our future work will mainly focus on the performance optimization of transformation algorithm and furthermore design the verification methods for other layers like R-layer, G-layer and P-layer.

Acknowledgment

This paper is partially supported by a NKBRPC (2007CB310803), the National High Technology Research and Development Program (863 Program) of China (2007AA01Z143), the National Natural Science foundation of China under Grant No. 60873118 and 60973147 and the Scientific Research Foundation of Beijing Jiaotong University under Grant No.2007RC110.

References

[1] Lamsweerde A V, "Goal-oriented Requirements Engineering: a Guilded Tour," In Proc. of the 5th IEEE International Sysmposium on Requirements Engineering, Toronto, Canada, 2001, pp, 249~263.

[2] Griss M L, Favaro J, d'Alessandro M, "Integrating Feature Modeling with the RSEB," In proc. of the 15th International Conference on Software Reuse, 1998, pp, 76-85.

[3] Falbo R A, Guizzardi G, Duarte K C, "An Ontological Approach to Domain Engineering," In: Proc of the International Conference on software Engineering and Knowledge Engineering (SEKE'02), LNCS 2503, 2002, pp, 167-181.

[4] Wang J, He K, Li B, Liu W, Peng R, "Meta-models of Domain Modeling Framework for Networked Software," In Proceedings of the Sixth International Conference on Grid and Cooperative Computing, Urumchi, China, August 2007, pp, 878-885.

[5] Wang J, He K, Gong P, Wang C, "RGPS: A Unified Requirements Meta-Modeling Frame for Networked software," In Proc of Third International Workshop on Advances and Applications of Problem Frames(IWAAPF'08) at 30th International Conference on Software Engineering(ICSE'08) ,Leipzig, Germany, 2008, 29-35.

[6] Marek Obito,"Ontologies and Semantic web: OWL DL Semantics," http://www.obitko.com/tutorials/ontologies-semantic-web/owl-dl-semantics.html, 2007.

[7] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider,The Description Logic Handbook: Theory, Implementation, Applications. Cambridge University Press, Cambridge, UK, ISBN 0-521-78176-0, 2003, pp, 1-99.

[8] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, et al, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML,"http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/, 2004-5-21.

[9] E.M. Clarke, O. Grumberg, and D. Peled, Model checking, MIT Press, 1999.

[10] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland," Business Process Execution Language for Web Services version1.1," http://www.ibm.com/developerworks/library/specification/ws-bpel/, 5 May 2003

[11] M. Ben-Ari, Principles of the Spin Model Checker, Springer Verlag, 2008, pp 29-44.