***J. Software Engineering & Applications*, 2009, 2:13-19**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

An Approach to Generation of Process-Oriented Requirements Specification

Jingbai Tian[1], Keqing He[1], Chong Wang[1], Huafeng Chen[1]

[1]State Key Lab of Software Engineering, Wuhan University, China
Email: Tianjingbai@live.cn

Received January 10th, 2009; revised February 21st, 2009; accepted March 19th, 2009.

ABSTRACT

In service-oriented computing, process model may serve as a link to connect users' requirements with Web Services. In this paper, we propose an approach and related key techniques to generate process-oriented requirements specification from user's goal. For this purpose, a requirements description language named SORL will be provided to capture users' requirements. Then, a unified requirements meta-modeling frame RPGS will be used to construct reusable domain assets, which is the basis of generating requirements specifications. Finally, a set of rules are defined to extract process control structures from users' requirements described with SORL, so that we can convert requirements description into process-oriented requirements specification smoothly.

Keywords*: *Requirements Specification*, *Process Modeling*, *Process Control Stucture*, *Networked Software

1. Introduction

Service-oriented computing (SOC) [1] paradigm is deemed as an active topic in recent years, because it provides a loosely coupled, highly interoperability and high levels of reuse application architecture. As a typical kind of SOC applications, networked software (NS) [2,3] is born as a software system whose structure and behavior can evolve dynamically. Following the idea of user-centric development, NS can satisfy users' demands dynamically by composing web services distributed among the network hosts.

When customizing personalized requirements of NS, one of the key problems is to elicit users' requirements precisely. In the network environment, it is difficult for users to describe their requirements correctly and completely. What's more, users' requirements are always with preference and rapid change. Traditional requirements elicitation methods collect initial requirements by memo of interview transcripts, which is hard to capture rapid changes of requirements in the network environment. Moreover, Users usually describe requirements with imprecise natural languages, and it's difficult for them to grasp formal or logical requirements description languages. Accordingly, a requirements description language called Service-Oriented Requirements Language (SORL) [4] was proposed in our previous works. It defined set of natural language patterns to describe requirements and supported online requirements elicitation.

In order to develop software rapidly, many previous researches emphasize on using domain acknowledge to support requirement assets reuse. In order to analyze and satisfy users' requirements in networked environment, we proposed a domain requirements metamodelling framework called RGPS (Goal-Role-Process-Service) [5]. It is tended to model common requirements assets in a domain, including domain ontology and four types of domain models. Domain ontology consists of entity ontology and operation ontology, which represents noun terms and verb terms respectively. Domain models can resides in role layer, goal layer, process layer and service layer. Role layer describes organizations, roles, participants and the relationships between them. Goal layer addresses users' requirements based on SORL and provides a goal based hierarchy for requirements refinement. Process layer and service layer present the processes and web services that can fulfill users' requirements.

Based on RGPS framework, users' requirements in SORL can be translated into customized goal model and process chain step by step. Finally, these models expressed with OWL can be saved as the corresponding Web service-based requirements specifications. During the process of translating, process model plays a very important role. Because process can not only give a detailed perspective on how a service operates, but serve as a bridge between users' requirements and service composition. According to the refining process in RPGS, goals can be refined as operational goals, which can be mapped to some certain processes. Compared with goal model, process model contains control structures. In order to generate complete process chain from initial requirements, a control structure extraction method is expected. In this paper, an approach to generate process-oriented requirements specification is proposed to extract control structure from language patterns and business rules. This approach can be used to generate a process-based requirements specification from users' requirements and support web services composition in NS.

The rest of this paper is organized as follows: Section 2 introduces the theoretical foundation of our approach. Section 3 gives the overview of the requirements specification generation approach. Section 4 states the process control structure extraction method with a case study. Section 5 discusses related works, followed by the conclusions and future work in Section 6.

2. Theoretical Foundation

2.1 SORL: A Requirements Description Language

Natural language pattern is the key to information extraction. Based on natural language patterns, SORL is designed for online requirements elicitation. Figure 1 illustrates the requirements description metamodel in SORL. There are four layers in the syntactic structure of SORL, which are sentence, pattern, phrase and word. More specifically, Word includes concepts in the domain ontology and key words defined in sentence and pattern (e.g. preposition and adverb).

There are 10 types of patterns in SORL, covering the description of functional goals, qualitative nonfunctional goals, quantitative non-functional goals, and precondition and post condition of a goal, etc.

The SORL sentence, which can be simple sentence or compound sentence, is composed of patterns. Simple sentence can be used to describe an activity, a constraint, or a state. Compound sentence is composed of natural language pattern, which can be *Loop Sentence*, *Choose Sentence* and *Trigger Sentence*. In this paper, we will pay more attention to compound sentences. More details of SORL are provided in [6].

2.2 RGPS: A Domain Metamodeling Frame

In order to develop software rapidly, reuse of both code and components should be taken into account. Common requirements resources are also important asserts in software development, which should be given to recovery and reuse. Many researches recognized the potential benefit of requirements reuse [7,8], and provided corresponding methods. In [9], for example, ontology was used to represent common requirements in certain domains. In order to sort and reuse requirements assets in the networked environment, a unified requirements meta-modeling frame named RPGS is proposed. The frame is consisted of domain ontology and domain model. Domain ontology includes entities and operations, which can be imported by domain models. There are four meta-models in RGPS:

Role metamodel describes the user roles of networked softwares, the actors who play roles, the contexts where actors are, intercourse between different roles, and the business rules to be complied with.

Goal metamodel describes the requirements goals of users, including functional, non-functional goal, and semantic relationships between them (e.g. depend, conflict, etc.). The goals can be refined to operational goal further, which can be realized by business processes. The process of goal refinement will not halt until all the goal leaves are operational goals.

Process metamodel defines information of business processes, including functional description such as IOPE (input, output, precondition and effect), and non-functional description such as *Qos expectation* and individualized *business context expectation*. A process can realize at least one functional goal and promote realization of non-functional goals. A process can be either an atomic process or a composite process, and a composite process has its own control structures.

Service metamodel provides the solution based on service resources, with functional and non-functional attributes. A service-based solution can be used to realize the specified business process.

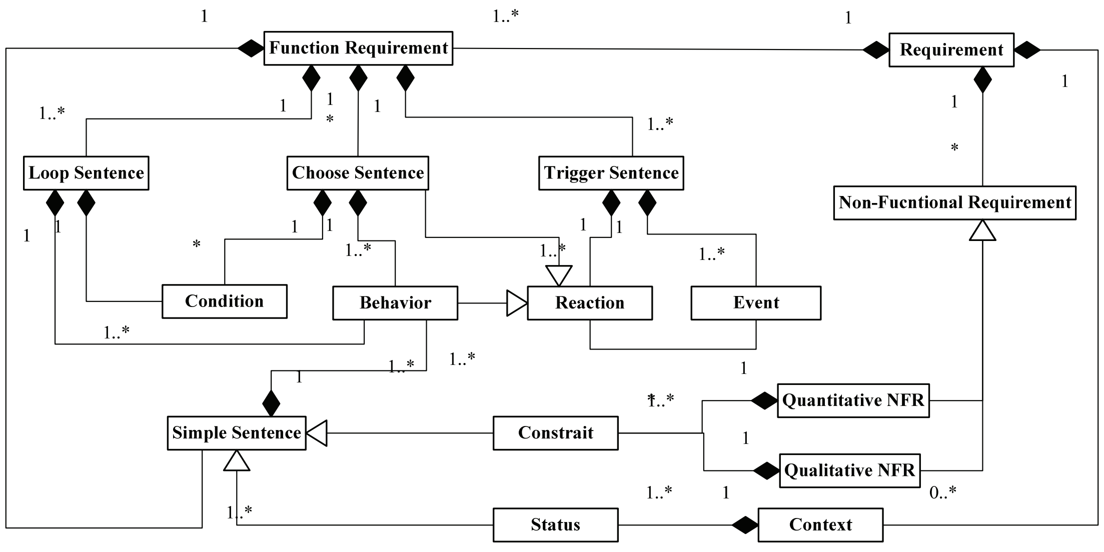

Figure 1. Requirements description metamodel

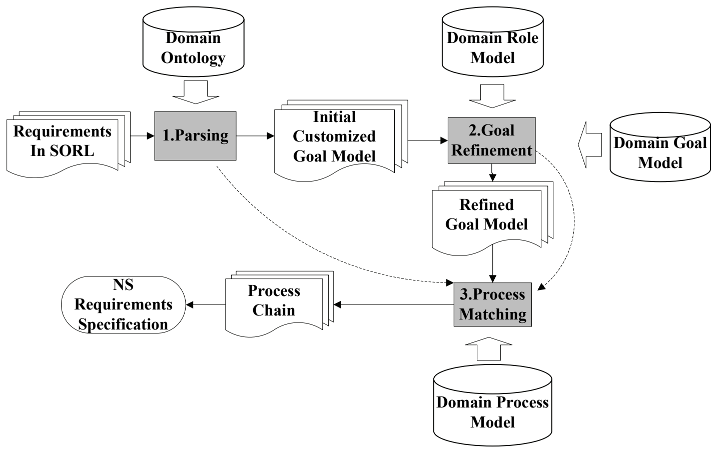

Figure 2. Overview of the NS requirements specification generation process

The domain model is provided by domain experts under the guidance of these four metamodels. Some domain requirements assets are stored as domain knowledge in each layer of RGPS, which corresponds to the respective requirements of different users group. Requirements of the user are elicited, analyzed, and finally matched to the common domain requirements specifications described with Web services.

3. Overview of the Approach

Figure 2 illustrates the overview of the requirements specification generation process. The three shadowed parts are key steps to convert requirements into NS requirements specifications.

- The first step is to parse users' requirements described in SORL to initial customized goal models by a finite automaton. SORL and domain ontology share a common vocabulary to ensure that the requirements can be parsed to goal model in a consistent manner. All the functional goals and non-functional goals are matched against domain goal model. All the unmatched goals will be marked, and stored into the specification repository for further analysis by domain engineers to check whether common requirements assets are enough.
- Secondly, each goal in the first step will be linked to a role. The goals can be decomposed into more concrete sub-goals according to the hierarchical structure in domain goal model. While the sub-goals are all operational goals, the refinement process is ended. The goal refinement process is shown in Figure 3. An operational goal means a goal which can be matched to a simple process or a composite process consistently. In this step, the goals depended by existing goals from users' requirements will also be added to improve users' demands. For example, if user needs the goal "book airline ticket by credit card". To achieve this goal, the goal "validate credit card" must be carried out previously.

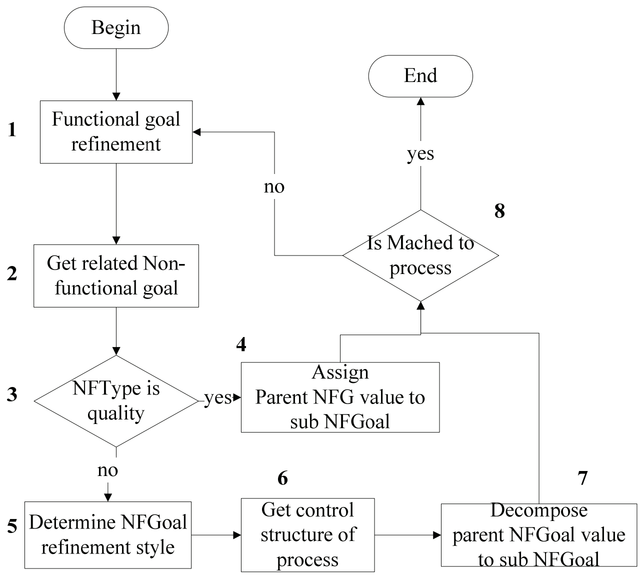
Begin
End
1
Functional goal refinement
yes
no
8
Is Mached to process
2
Get related Non-functional goal
4
3
NFType is quality
yes
Assign Parent NFG value to sub NFGoal
no
6
7
5
Determine NFGoal refinement style
Get control structure of process
Decompose parent NFGoal value to sub NFGoal

Figure 3. Goal refinement process

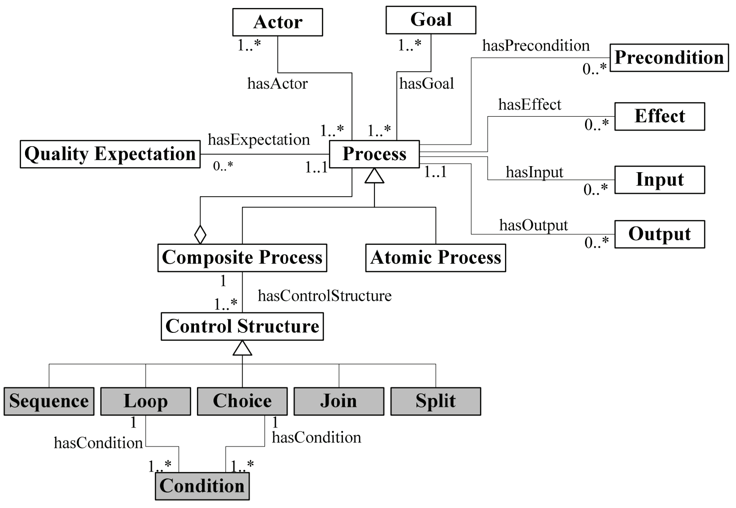

Figure 4. Process metamodel in RPGS

- In the third step, a set of processes are gained based on the refined goal model composing operational goals. Since process model has its own control structure (e.g. sequence, choice, join, etc), a set of extraction rules is needed to extract control structures from above two steps and composite these processes into process chains. With the control structures, the discontinuous processes can be integrated as process chains and saved as requirements specification finally.

In this section, the overview of our approach is illustrated, in next section, the process control structure extraction rules will be discussed in detail.

4. Extraction Rules for Process Control Structure

This section discusses some rules of extracting process control structures. Figure 4 illustrates the process metamodel in RPGS frame. The shadowed parts are primitive control structures defined in RPGS, i.e. Sequence, Loop, Choice, Join, and Split. In order to obtain control structures in process chains, each control structure should be mapped in metamodel layer.

In our approach, process control structures are generated from two sources: one is sentences based on SORL. The other is Depend relationships between related goals of the specific process.

4.1 Extraction Rules Based on SORL Sentence

As Section 2 describes, two kinds of compound sentence in SORL can be used for process control structure extraction, i.e. Loop sentence and Choose sentence.

Loop Sentence

The loop control structure can be extracted from loop sentence. Loop sentence describes system cyclic behavior under a certain condition. As Figure 5 shows, loop sentence is represented as "loop <function requirement pattern> until <condition pattern>". To extract loop control construct from the corresponding loop sentences, we define extraction rules as follows:

Def.1 In a Loop sentence, *FP* is the Function Requirement Pattern and *CP* is the Condition Pattern. So the Loop sentence "loop *FP* until *CP* is true" denoted as *LoopSentence* (*FP, CP*).

Def.2 In a process chain, *P* is a process and *C* is a condition. "Loop a Process *P* until condition *C* is true" will be denoted as *LoopControlStructure* (*P, C*).

Def.3 If *FP* is a Function Pattern, the process related to the corresponding goal of this Function Pattern is denoted by *P*. If *CP* is a Condition Pattern, the related Condition in process chain is denoted by *C*.

Extraction Rule.1:

$LoopSentence\,(FP,CP) \Rightarrow LoopControlStructure\,(P,C)$

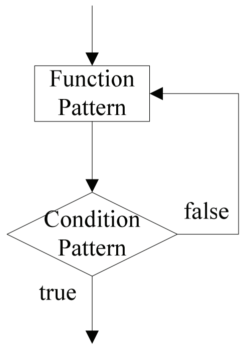

Figure 5. Loop sentence

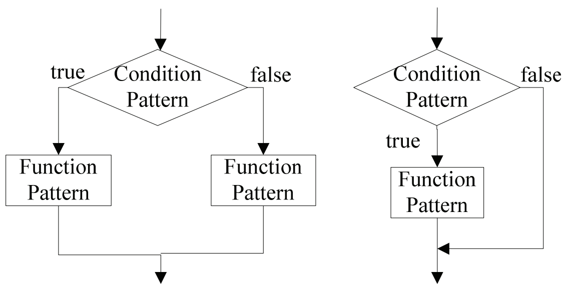

Figure 6. Choose sentence

Choose Sentence
The extraction of Choice control structure is similar to the one of Loop control structure. In SORL, choose sentence describes a selection of different system behaviors. Choose sentence is represented as "if <Condition Pattern>, then <Function Pattern>, else <Function Pattern>", as Figure 6 depicts. The extraction rules of choice control structure are the followings:

Def.4 FP_A and FP_B are two alternative Function Requirement Patterns in a Choose Sentence, and *CP* is the Condition Pattern. The Choose sentence "if *CP*, then FP_A, else FP_B" is denoted by *ChooseSentence* (*CP*, FP_A, FP_B). When *FP* is a Function Requirement Pattern in a Choose Sentence, the Choose sentence "if CP, then FP" is denoted by *ChooseSentence* (*CP*, *FP*).

Def.5 *C* is a condition in process chain, and P_A and P_B are two processes. The Choice control structure in process chain is denoted by *ChoiceControlSturcture* (*C*,P_A,P_B), which means "if C is true, then execute PA; if not, execute PB". When P is a process, if C is true, then executes P, else skip P, this is denoted by ChoiceControlSturcture (C,P)

Extraction Rule.2:
ChooseSentence (*CP*, FP_A, FP_B) $\Rightarrow$ *ChoiceControlStructure* (*C*, P_A, P_B)
ChooseSentence (*CP*,*FP*) $\Rightarrow$ *ChoiceControlStructure*(*C*,*P*)

4.2 Extraction Rules based on Goal Model

In RGPS, Goal metamodel defines Depend relationships between goals. A Depend relationship can be either an Object Depend or a Conditional Depend. Object Depend used to indicate the Input/Output relationship between two goals, which is related to dataflow in system. Conditional Depend describes the Precondition/Postcondition between two goals, which is related to business rules.

Object Depend
In Object Depend relationship, realization of a goal depends on output data of the other goals. We assume that such dataflow in real time systems would never get failure. Correspondingly, the realization of related process also depends on the others. Therefore, we can extract process control structure from Object Depend relationship this section defines three extraction rules for Sequence, Split and Join control structure respectively.

Def.6 G_A and G_B are two goals in refined goal model. If G_A Object Depend on G_B, the relationship is denoted by *ObjectDepend* (G_A, G_B). $G_{Ai\ (i=1,\ 2,\ldots,n)}$ and G_B are goals in refined goal model, $G_{Ai\ (i=1,\ 2,\ldots,n)}$ Object Depend on G_B, it is denoted by $\underset{i=1,2\ldots n}{And}\ objectDepend(G_{Ai}, G_B)$. G_A and $G_{Bi(i=1,2,\ldots,n)}$ are goals in refined goal model, G_A Object Depend on $G_{Bi(i=1,\ 2,\ldots,n)}$, it is denoted by $\underset{i=1,2\ldots n}{And}\ objectDepend(G_B, G_{Ai})$.

Def.7 P_A and P_B are two processes in process chain. If there is a sequence control structure between P_A and P_B and P_A executes after P_B, denoted by *Sequence* (P_A, P_B).

Extraction Rule.3:

$$ObjectDepend(G_A, G_B) \Rightarrow Sequence(P_B, P_A)$$

Def.8 $P_{Ai(i=1,\ 2\ldots\ n)}$ and P_B are Process in process chain, if $P_{Ai(i=1,\ 2\ldots\ n)}$ is split from P_B, the control structure is denoted by *Split* (P_B, P_{Ai}).

Extraction Rule.4:

$$\underset{i=1,2\ldots n}{And}\ ObjectDepend(G_{Ai}, G_B) \Rightarrow \underset{i=1,2\ldots n}{And}\ Split(P_B, P_{Ai})$$

Def.9 $P_{Ai(i=1,\ 2\ldots\ n)}$ and P_B are Process in process chain, if $P_{Ai(i=1,\ 2\ldots\ n)}$ is join to P_B, the control structure is denoted by *Join* (P_{Ai}, P_B).

Extraction Rule.5:

$$\underset{i=1,2\ldots n}{And}\ ObjectDepend(G_B, G_{Ai}) \Rightarrow \underset{i=1,2\ldots n}{And}\ Join(P_{Ai}, P_B)$$

Conditional Depend
Conditional Depend describes the business rules between goals. "Goal A is Conditional Depend on Goal B" means "if Goal B cannot be achieved, then Goal A cannot be achieved; if not, it should be skipped". For example, a traveler want to travel to Wuhan, he want to "book an airline ticket to Wuhan", and to "book a standard room in Wuhan". If the first goal can't be achieved, which means he can't arrive in Wuhan in time, the second goal will be canceled. From this point of view, we defined the following rules:

Def. 10 G_A and G_B are two goals in refined goal model, If G_A Conditional Depend G_B, denoted by *ConditionalDepend* (G_A, G_B).

Def. 11 P_A is a process, if P_A is successful execution, the condition C_A in P_A 's Effect would be True, else be False.

Extraction rule.6

$$ConditionalDepend(G_A, G_B) \Rightarrow ChoiceControlStructure\ (C_B, P_A)$$

4.3 Case Study

In this section, a case in travel and transportation domain will be demonstrated. Firstly, we constructed the domain ontology and domain model in [5]. And the following ones are requirements examples.

A customer is planning a trip to Wuhan. His SORL-ased requirements are as followings:"Query travel expense in Wuhan", "Book an airline ticket to Wuhan by credit card", "if the price is no more than 500 RMB, then, book a room, the standard of the hotel is no less than 4-Star, else, book a standard room, the standard of the hotel is equal to 3-Star."

Figure 7 Refined goal model of the case.

- Based on extraction rule 2, we can find that there is a choice control structure in the corresponding processes of "book room".
- Based on extraction rule 3, the related process of "Validate Credit Card" must be executed before the related process of "Book Airline Ticket by Credit Card".
- Based on extraction rule 6, if "book airline ticket" cannot be achieved, "book room" should be skipped.

The generated process chain is shown in Figure 8.

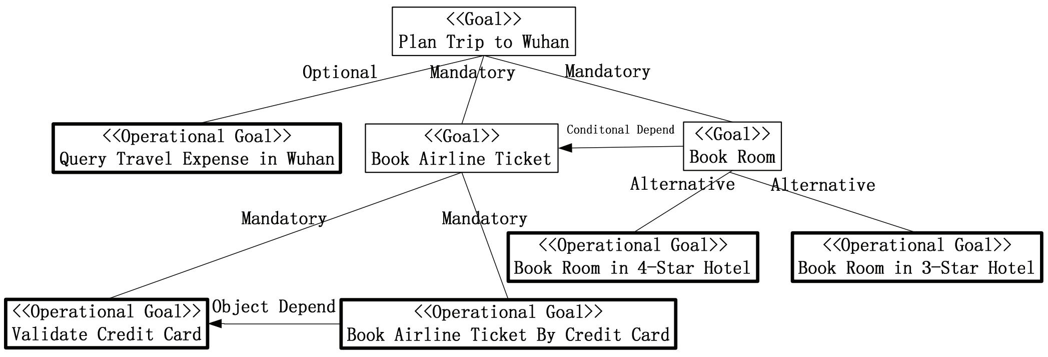

Figure 7. An example of refined goal model

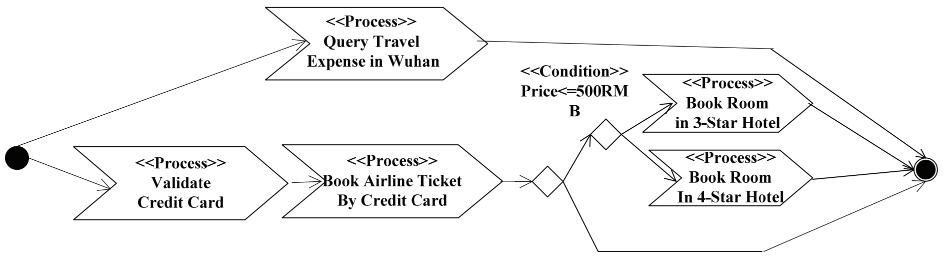

Figure 8. An example of process chain

A trial system based on this approach is build up preliminarily. With the graphical process tool, users can see their requirements directly, so that users can understand the work flow intuitively and modify the work flow with ease.

5. Related Works

Process-oriented technology has many potential advantages in software development, and many SOC technologies have adopted process in their methods to improve the service composition. For example, OWL-S [10] models services as processes to give services a more detailed perspective. In WS-BPEL [11], processes use Web Service interfaces to export and import functionality.

In [12], W. M. P. van der Aalst proposes an algorithm based on petri net to extract process model from “workflow log”, which contains information about the workflow processes in real-time systems. Moreover, many systems (e.g. ERP, CRM) provide a set of pre-defined process models. In our work, we use RGPS to manage not only process model but also related goal, role and service model. A solution for process model generation from user’s requirements is given.

In [13] and [14], authors established relationships between process model and high-level requirements models (e.g. KAOS and i*), presented methods to generate, validate and improve process model. In [15], authors present a framework and associated techniques to synthesis service composition from temporal business rules through process models. Compared with these approaches, our approach is more suitable for end-users and caters for the “user-centric” trend.

6. Conclusions and Future Work

In this paper, we proposed an approach to generation of process-oriented requirements specifications. It establishes mappings from SORL metamodel and Goal metamodel in RGPS to the control structures defined in Process metamodel. So it can be used to generate an NS requirements specification including goal models and process chains through these control structures.

In the future, the control structure extraction rules will be improved, the influences from goal refinement types (e.g. mandatory, optional) to the control structure will be supplemented. Moreover, the corresponding validation and verification methods will be provided later, followed by the relevant tools and platforms.

7. Acknowledgements

This research project was supported by the National Basic Research Program of China (973) under Grant No. 2006CB708302 and 2007CB310801, the National High Technology Research and Development Program of China (863) under Grant No.2006AA04Z156, the National Natural Science Foundation of China under Grant No. 60803083, 60873009, 60803025 and 60703018, the Provincial Natural Science Foundation of Hubei Province under Grant No.2006ABA228,and the Research Fund for the Doctoral Program of Higher Education of China under Grant No. 60803025.

REFERENCES

[1] M. N. Huhns and M. P. Singh, “Service-oriented computing: Key concepts and principles,” IEEE Internet Computting, pp. 2–8, 2005.

[2] K. He, R. Peng, J. Liu, F. He, *et al.*, "Design methodology of Networked Software Evolution Growth based on Software Patterns," Journal of System Science and Complexity, Vol. 19, pp. 157–181, 2006.

[3] K. He, P. Liang, R. Peng, *et al.*, "Requirement emergence computation of networked software," Frontiers of Computer Science in China, 1(3), pp. 322–328, 2007.

[4] W. Liu, K. He, J. Wang, *et al.*, "Heavyweight semantic inducement for requirement elicitation and analysis," In Proceedings of the 3rd International Conference on Semantics, Knowledge and Grid (SKG 2007), Xi'an, China, pp. 206–211, 2007.

[5] J. Wang, K. He, P. Gong, C. Wang, R. Peng, and B. Li, "RGPS: A unified requirements meta-modeling frame for networked software," in Proceedings of Third International Workshop on Advances and Applications of Problem Frames (IWAAPF'08) at 30th International Conference on Software Engineering (ICSE'08), Leipzig, Germany, pp. 29–35, 2008.

[6] L. Wei, "Research on services-oriented software requirements elicitation and analysis," Ph. D thesis, Wuhan University, 2008.

[7] P. Massonet and A. van Lamsweerde, "Analogical reuse of requirements frameworks," Third IEEE International Symposium on Requirements Engineering, pp. 26, 1997.

[8] W. Lam, "A case-study of requirements reuse through product families," Springer Netherlands, pp. 253–277, 1998.

[9] Y. Q. Lee and W. Y. Zhao, "Domain requirements elicitation and analysis–An ontology-based approach," In: Proceedings of Workshop on Computational Science in Software Engineering (CSSE), pp. 805–813, 2006.

[10] M. Smith, C. Welty, and D. McGuinness, "OWL web ontology language guide," W3C Candidate Recommendation, 2004. Available: http://www.w3.org/TR/owl-guide/.

[11] OASIS: Web services business process execution language version 2.0, 2006. Available: http://docs.oasis-open.org/ wsbpel/2.0/wsbpel-specification-draft.html.

[12] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," IEEE Transactions on Knowledge and Data Engineering, Volume 16, Issue 9, 2004.

[13] G. Koliadis and A. Ghose, "Relating business process models to goal-oriented requirements models in KAOS," PKAW 2006, pp. 25–39, 2006.

[14] A. Ghose and G. Koliadis, "Actor eco-systems: From high-level agent models to executable processes via semantic annotations," COMPSAC 2007, pp. 177–184, 2007.

[15] J. Yu, Y. B. Han, J. Han *et al.*, "Synthesizing service composition models on the basis of temporal business rules," Journal of computer science and technology 23(6) pp. 885–894, 2008.