Ontology-Based Exception Handling for Semantic Business Process Execution

Kai Zhao[1,2]
[1]School of Information Science and Engineering, Xinjiang University, Urumqi, China
Email: zhawkk@xju.edu.cn

Linlin Zhang[1] and Shi Ying[2]
[2]State Key Laboratory of Software Engineering (Wuhan University), Wuhan, China
Email: zllnadasha@xju.edu.cn, yingshi2968@yahoo.com.cn

***Abstract*—Along with the widespread acceptance of business process management (BPM) and Semantic Web services composition technologies, Semantic Web service oriented programming is becoming an efficient way to develop modern business applications. As Semantic Web services are inherently unreliable, how to develop reliable service oriented applications is a significant and challenging problem, especially in complex, untamed and dynamic services environment. However, current business process programming languages for Semantic Web services provide almost no support for exception handling, and runtime environments are weak in providing reliability and adaptability. In this paper, we propose an ontology based exception handling method for semantic business processes using Semantic Programming Language (SPL). We identity a new exception taxonomy described as exception ontology and devise an event-driven exception events detection mechanism. We also devise an exception handling framework based on Multi agent system for SPL program. Especially, the framework provides forward recovery support with dynamically substituting failed services when an exception arises during execution by semantically equivalent or semantically similar Web services.**

***Index Terms*—Exception handling, Semantic Web services, Ontology, Semantic Programming Language, Agent**

I. INTRODUCTION

At the crossing of the semantic Web and Service Oriented Computing domains, research in the field of semantic Web services [1] and related program development approaches is very active. The semantic Web and semantic Web service technologies have already attracted significant attention during the last decades, while these technologies are now becoming a well-established branch of software engineering. The proliferation of semantic Web services standards reflects the industry interest and demand for distributed enterprise applications. Depending on such loosely coupled components with rich semantic descriptions makes it easier to develop, maintain, and modify the applications. Even though, issues of exception handling, recovery and adaptation are still far from being extensively studied in areas such as semantic Web service-oriented programming languages, workflow systems, especially in semantic business process management systems [2]. Traditional approaches to exception handling in workflow and business process management rely on exception handling primitives built into the program language. For example, WS-BPEL [3] provide basic exception handling activities, such as fault handlers and compensation handlers, then programmers can specify process control logic with exceptions and perform exception detection, handling and propagation. However, exception handling mechanism in WS-BPEL does not provide any explicit and direct support for semantic Web services-based software systems.

Current standards and proposals for semantic Web services (such as OWL-S[4], WSMO[5], SAWSDL[6] and BPEL4SWS[7]) enhance Web service standards with rich semantic annotations to facilitate dynamic discovery, invocation and composition resiliently, but provide partially, even not support for above mentioned exception handling problems. In this paper, we focus on presenting a systematic method to handle failures of semantic application level during execution of semantic Web services-based software systems. Specifically, we consider Semantic Programming Language (SPL) [8] and illustrate automatic handling of semantic application level exceptions in SPL programs, but we believe our method has a great potential for the reliable execution of semantic Web services-based software systems and has benefits in a way that is independent from SPL.

Our main contribution revolves around a proposal for augmenting SPL with fault handlers so that programmers can specify a reliable, possibly adaptive program using SPL. This proposal consists of: (1) an augment for SPL with fault handlers, which handle exception conditions with appropriate semantic information that aid to resolve the exceptions; (2) an exception ontology built on the notion of semantic information to support explicit description of exception conditions; (3) a Multi Agent System (MAS) [9] based framework for the reliable execution of SPL programs, which handle exception conditions using exception ontology and Web services' contexts. Our proposal allows for a dynamic detect exception conditions and modify the control flow of SPL

doi:10.4304/jsw.7.8.1791-1798

program at runtime in the presence of exceptions. As long as an exception happen, our proposed framework reason about semantic equivalence or most semantic similarity of services and apply a forward failure recovery for substituting a semantically equivalence or at least semantically similar service for the failed service by find, diagnosis and select dynamically.

There are some basic assumptions of our method, including: (1) exception conditions, which encounter during execution of a semantic Web services-based application, have some special characteristics with respect to traditional Web services-based applications. Despite the fact that some of the programming languages for Web services have a wide adoption, such as WS-BPEL, which act as the de facto standard for Web services, these languages still remain focused on the syntax aspects without emphasis on the semantic level. Many approaches of exception handling are not initially considered to meet the peculiar requirements of a successful semantic exchange. Semantic exchange makes partners understand the content of messages what the send and receive. However, these approaches do not exploit the potential that exception information offers when it comes to describing the different aspects of exception data for effectively exception handling. (2) Existing lots of Web services that are semantically equivalent and/or similar in dynamic Web environment. Due to the fact that more and more enterprises and organizations can provide Web services to their business partners freely so that we can speculate that some Web services might have exactly the same and/or similar functionality while still differ in non-functional properties.

On the basis of SPL, in this paper, we propose a novel and flexible framework to handle and resolve semantic exceptions occurring in the execution of SPL programs. The rest of this article is organized as follows. Section II introduces the classification of exception and language foundations of our method in detail. Section III presents our framework based on MAS and describes how this framework has been implemented. Section IV summarizes related work. Section V outlines our plans for future works and concludes.

II. Exception Handling And Semantic Programming Language

A. Semantic Programming Language

Semantic Programming Language (SPL) is an extension of WS-BPEL that facilitates to orchestrate semantic Web services, traditional Web services and an admixture of them. A XML-style SPL program which has been transformed and deployed successfully can be exposed both as semantic Web services and conventional Web services. SPL extends WS-BPEL via defining a mechanism to attach the semantic descriptions on the activity level and process level using RDF4S [10] ontologies. Therefore, SPL enables semantic Web service-oriented program development and semantic business process-driven application integration. To developing application at semantic layer based on ontologies, SPL introduces five new constructs which decouple from WSDL: (1) The Semantic Data Type. It defines data types with specific semantic information which is described by ontological concepts. (2) The Semantic Variable. It defines the semantic data structures of a SPL program according to Semantic Data Types. It provides the means for holding semantic messages that interacting with external services, and also hold state data that are related to the business processes and business rules. (3) The Semantic Service. It describes the external partners' services using semantic information (classification semantics, data semantics, functionality semantics etc.) based on the semantic Web services ontology. The definition of semantic service consists of a collection of semantically defined messages and operations, thus they abstractly represent the partner services that cooperate to fulfill the program requirements. (4) The Semantic Rule. It semantically defines the business decisions, policies and constraints on the behavior of business process. (5) The Semantic Process. It defines the process procedures logic, composed from activities (for Semantic Web Services). The definition of semantic business process is a model for describing the behavior of a business process based on interactions with external semantic Web services. The main SPL constructs and activities are summarized in Table I.

TABLE I.
SUMMARY OF SPL SEMANTIC ACTIVITIES

Name	Description
invoke	Semantic Web service call.
receive	Wait for a Semantic message to arrive.
assign	Manipulate Semantic variables.
reply	Respond to a remote operation.
call	Semantic sub-process call.
wait	Delay execution for a while or deadline.
throw	Indicate occurrence of Semantic exception.
try-catch	Capture and handle Semantic exception.
terminate	Terminate a Semantic Process instance.
Structured Semantic Activities	
Activity	Description
sequence	Execute activities in sequential order.
if	Selection an execution path with condition.
while	Repeat execution a code block
parallel	Concurrent execution

SPL makes use of semantically annotated data types to enhance data processing by means of ontology-based semantic data adaption process. That is, SPL directly use ontologies as data model. Activities implementation in SPL can be described using other SWS frameworks such as OWL-S or WSMO. As the first class citizen of SPL, Semantic Service that offers functionality to user can also be defined using other SWS frameworks. SPL uses RDF4S ontologies as data model and annotates data types by referring to ontological concepts directly. With the hierarchy of ontology concepts, an implicit semantic data flow has been introduced in SPL via Semantic Data Type & Variables. Semantic Variable can be defined either global or local (scope). Then, different semantic activities

in SPL program can use these Semantic Variables as input/output containers and access semantic data respectively. Additionally, Semantic Data can be duplicated from one variable to another.

In particular, the exception handling mechanism of SPL is based on a standard notion of explicit fault handlers known from programming languages such as WS-BPEL. For well understanding and using the content of semantic exception messages, SPL explicitly describe the semantic information of exception and other related information, and aim at propelling exception handling to the level of semantic. The exception handling of SPL provides flexibility and reliability for SPL program to meet the requirements of a semantic exception processing in SPL based on the runtime support environment.

B. Exception Detection Knowledge

As mentioned earlier, SPL remains focused on the semantic aspects of composition of Web services. Business processes-centric applications specified using SPL will interact with their partners through operation invocations of semantic Web services using ontologies as agreements on a common vocabulary. Semantic Web services are still based on loosely coupled Service Oriented Architecture (SOA). The communication between services is done over Internet connections that may or may not be highly reliable. Semantic Web services could also raise faults due to logical errors and execution errors arising from defects in the infrastructure.

By analyzing exception knowledge of SPL program, exception pattern and exception transmitting mechanism, we fuse Semantic knowledge and ontology technology. Based on the decomposition of exception knowledge, we construct exception ontology from the top down based on analyzing exception process of SPL program and the logic relation between exception reasons and symptom in complex Internet environment. The exception knowledge base level structure is shown in Fig. 1. The domain layer illustrates the top concepts of SPL exception knowledge. Class layer classify all concepts with relation of category and inheritance. Property layer includes concepts and property information. Instance level includes information of instance on the relevant property concepts.

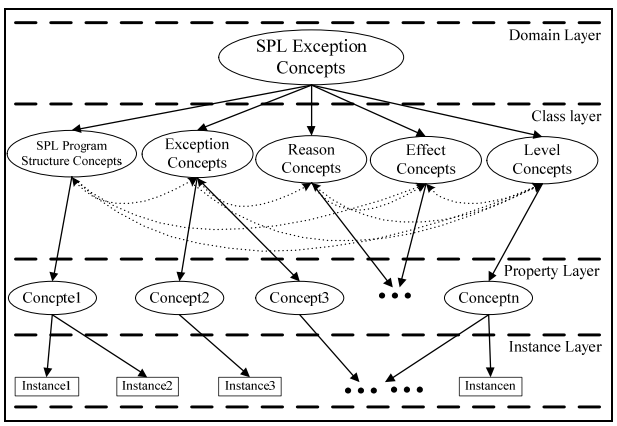

Figure 1. Level structure of SPL exception knowledge ontology.

SPL programs will need to handle exceptions appropriately. SPL programs may also need to signal exceptions themselves. Exception detecting, handling and signaling is an important aspect of programs designed using SPL. For the purpose of handling exceptions effectively, we clarify the conditions under which exceptions may occur during the execution of SPL programs. Based on the exception knowledge base level structure, exceptions in SPL can arise be distinguished three main categories, see Fig. 2.

- **Service Invocation Exception:** exceptions are related to the invocation of services. For example, communication error, service unavailable, no response, malformed results, time out etc. These exceptions are driven by the improper execution or the unsuccessful logical outcome of external services execution. In addition, when external services complete, but the predefined SLA properties have been degraded due to various reasons, so an exception will be notified.
- **Semantic process runtime exception:** include all erroneous states specific to the application logic of a SPL program. Exceptions are driven by the inconsistencies or discrepancies on the semantic process model in SPL programs that may occur during the execution, and predefined the known or declared exceptions on the interfaces of semantic Web services, namely semantic application exceptions or user defined exceptions. When designing a SPL program, it's responsibility of programmer to specify the different exceptions a SPL program can throw so that exception can be appropriately catch and handle.
- **Semantic process model processing exception:** exceptions caused by parsing/syntax level problems and problems with malformed SPL program or ontologies files. For example, Semantic services or data transformation errors, Semantic rules processing failure etc.

These categories are application independent and are specified in the exception ontology, so different kinds of exceptions raised during execution of SPL program can be recognized or identified during the design and runtime phases with domain ontologies and reason engine support.

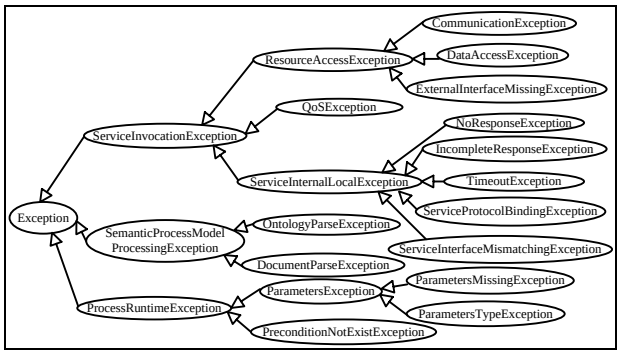

Figure 2. SPL exceptions classification.

C. Exception Handling Constructor

The exception handling approach of SPL is based on a standard notion of explicit fault handlers known from Java, C#. For well understanding and using the content of semantic exception messages, SPL explicitly describe the semantic of exception and other related information, and aim at propelling exception handling to the level of semantic. Such approaches use exception ontology to meet requirements of a successful semantic processing.

SPL has built-in exception handling constructor that can be used to build reliable business process driven by Semantic Web services. Every business process can define a list of fault handlers to respond to exceptions and to perform possibly recovery. A fault handler has the following form using the XML syntax, see Fig. 3:

It is same with WS-BPEL, the built-in construction element of exception handling in SPL is based on the notion of scopes. SPL provides fault handlers for trying to capture, deal with and recover from exception situations. SPL relies on exception ontology to describe the SPL exception classification, context, and other information. It introduces the <semanticExceptionType> and <semanticExceptionVariable> elements, see Fig. 4. They facilitate defining of semantic application exceptions and thus enable exchange complex exception message between SPL program and the runtime supporting environment. Fig. 4 helps illustrate some details of the semantic exception types and semantic exceptions variables in SPL with XML form. With regard to the syntax and usage of fault handlers in SPL, More details of can be found in [8].

```
<try-catch name="NCName">
    <try>      activity+      </try>
    <catch exceptionType="QName"?  exceptionIdentifier? >*
              activity*
    </catch>
    <catch exceptionName="QName"?
          exceptionVariable="VariableName"?
          ( Type="QName" | exceptionType="QName" )? >*
              activity*
    </catch>
    <catchAll>?      activity*      </catchAll>
    <finally>?     activity+  </finally>
</try-catch>
```

Figure 3. SPL exception handling syntax structure.

```
<semanticExceptionType name="noServiceFoundST"
   ontologyReference="onto4t#ServiceUnavilableException"/>
<semanticExceptionType name="engineNotExistST"
          ontologyReference="onto4t#BusinessException" >
   <semanticElement name="TimeStamp" SType="TimeST"/>
   <semanticElement name="Model" SType="ModelST"/>
   <semanticElement name="DetailDesc" SType="DescST"/>
</semanticExceptionType>

<semanticExceptionVariables>
     <semanticExceptionVariable name="err1"
          exceptionType="noServiceFoundST"/>
     <semanticExceptionVariable name="err2"
          exceptionType="engineNotExistST"/>
</semanticExceptionVariables>
```

Figure 4. Example of semantic exception type and variable.

When an exception occurs, the regular process control flow is interrupted and moves from the activity or sub-process that raises the exception to the interrelated scope's fault handler through semantically reasoning on captured exception. If there is no suitable scope level fault handler, then the execution moves to the process' fault handler. A fault handler can have many activities or have one activity that can be a structured activity. Once a fault handler is triggered to execute, all contained activities of the related scope are forced to terminate. Generally, once an exception arises, it is considered that the scope has not completed normally and is required for compensation or recovery. Exceptions are not handled by any explicit fault handler will resolved by an implicit default fault handler. This fault handler is always provided to catch all exceptions except those for which without any fault handler is attached.

D. Semantic Event Detection

SPL program contains rich Semantic information to further facilitate resilient dynamic Web service discovery, binding, invocation and execution. Since all these tasks are expected to be performed in dynamically changing environments with semi automatically, during execution, many practical exception or unusual problems are prone to arise. Execution infrastructure must able to effectively monitor and detect the exception, so that it can be able to interpret the reason and the effects of the exception, and support to deal with exception states. How to detect SPL program exceptions is an important task. Through exception detection, it can trace the sources of exceptions so that program can rescue, and avoid the conditions next time if possible.

To solve the problem of exception detection during the SPL program execution, an event-driven exception monitor and detection method is proposed, and two questions have been resolved: first, what exactly should be monitored and detected; second, what implementation method should be chosen. For the first question, by analyzing the structure and semantics of SPL program, it is possible to identify important events with relation to exceptions which might be detected. These event types that occur during the execution of the SPL program are summarized as follows:

- Service invocation. This type may be the most important event type. A service call by invoke activity can be identify as one event together with its inputs, outputs and effects parameters.
- Sub-process call. Similar as Service invocation type, sub-process call can be recorded as on event.
- Input assignment. Semantic data of input can be provided either by the data binding or user. Usually performed by assign activity.
- Output processing. Semantic data of output are obtained as a result of the Semantic Web service invocation. Usually performed by assign activity.
- Preconditions evaluation. Evaluates the preconditions of the activity (especially invoke activity) with Semantic data assigned and with the true or false.

- Effect evaluation. Evaluates the effect of the activity with Semantic data assigned and with the true or false.
- Primitive activity execution. It defines primitive activity (except invoke, assign, call activity etc.) execution events, such as reply activity.
- Structure activity execution. This type event associated with structure activity such as if, while, represents and captures its start and one its end.
- Exception event. It defines different categories of exception specific event types. Actually, exception knowledge is part of the event ontology.

It is important to note that presented event types are neutral to the application domain, so it can implement easily by SPL runtime environment. On the basis of this analysis, a hierarchy of event types is proposed; we construct events ontology to serve as a sound basis for the detection of SPL exception. The SPL runtime environment adopted the event-driven model to realize the exception detection. During the SPL program execution, SPL runtime environment emits exception events specific to the SPL program instance execution. These emitted exception events are instances of exception types defined in the events ontology. The content of raised exception event instances describes the context information once the exception occurred. The content is semantically annotated by the domain ontology concepts and events ontology concepts; it enables more flexible exception event detection techniques than those syntactic key-words matching. And more importantly, based on matching exception event type and related content, a Semantic reasoning for detecting exception events can be employed, also, the defined types can be used in exception handler as part of parameters.

III. ONTOLOGY-CENTERED, MAS BASED FRAMEWORK

As shown in Fig. 5, in order to handle all the semantic exceptions classified in the previous sections, we propose a exception handling framework based on MAS to fulfill the design principles for reliable executing SPL program. In this section we describe in detail the elements of the framework. The architecture that constitutes framework, which bases on technology of agent, ontology and semantic web services, consists of three main components: a set of agents that constitute MAS, four ontology libraries and an ontology reasoner. Our proposed framework incorporates a reasoner, which is central to the framework. All the agents use the ontology reasoned and are partly defined using the ontology repository. This module is able to deal with basic concepts, attributes and translate SWS descriptions.

A. Multi-Agent System

The core brain of our proposed framework is a Multi-Agent System (MAS). Its architecture consists of a group of soft agents that perform SPL-based semantic business process transformation, dynamic discovery & coordination, semantic service & data mediation, business rule execution, service dynamic binding and invocation and semantic exception handling.

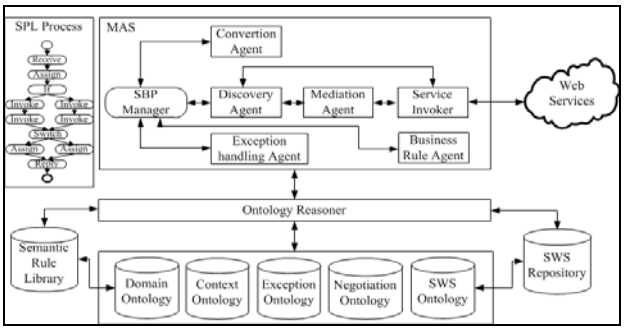

Figure 5. The exception handling framework based on MAS.

- The Semantic Business Process Manager (SBP Manager) is responsible for decoupling the static relation between business process and its external participants, converting the semantic information of business processes into the messages for facilitating the other component agents, triggering instance of semantic business processes, and monitoring and controlling the execution.
- Transformation Agent is responsible for parsing SPL program in accordance with the semantic rules of SPL, extracting service's semantics which is used as query conditions for Web services, mapping semantic Web services with appropriate Web services, and finally converting source program to executable semantic business process.
- Discovery Agent is in charge of performing discovery in the semantic web services repository for the set of services that satisfy the requisites established by the process and coordination between agents and web services. Such an agent performs very complex reasoning tasks, including: interpreting web service semantic queries; finding the best provider based on semantic queries; interacting with the service provider as necessary to fulfill the query and returning query results.
- Mediation Agent is responsible for specifying which semantic service and external web service are connected and which type of mismatches can be resolved between them.
- Service Invoker is responsible for invoking the selected web service on the process' behalf and returns the results to the semantic business process.
- Exception handling Agent is responsible for intervene in the default control flow of a SPL program's execution as soon as an exception has happened by modifying its process specification so as to get a semantically equivalent and to successfully continue execution using the substituted process version.
- Semantic Business Rule Agent is responsible for determining which rules of semantic business process could be trigger and how to trigger these rules. This agent can dynamically change status of SBP through operating variables of facts and return the results to SBP Manager.

Accomplishing these tasks requires ontologies to describe Web-service capabilities, interaction patterns, and domains along with a logic that allows reasoning on the information specified in those ontologies.

B. Ontologies Repositiory for MAS

Ontologies are the paramount technology as they operate as the 'glue' for the interoperation and integration of heterogeneous system: ontologies function as universal vocabularies so that Semantic Web services and agents share the same interpretation of the concepts contained in the exchanged messages; ontologies are useful to semantically describe web service capabilities and processes. Our method builds on the base of ontologies, namely ontology-centered, as Fig. 6 shown. From the agents' perspective, their local domain-related knowledge may be extracted from the application domain ontology. Ontologies composes importantly epistemological basis for the proposed framework. Motivated by the design principles, these ontologies provide semantic relationships between the knowledge level components describing semantic Web services and the conditions related to its use, which also part of the RDF4S. On the other hand, in order for Agents to successfully accomplish their assigned tasks, those ones also provide various data repositories containing the knowledge that is necessary to perform the assignment.

In particularly, among these ontologies, we propose exception ontology and context ontology for the purpose of diagnosing and handling exceptions. These ontologies facilitate the unambiguous interactions between agents during an exception resolving process. The exception ontology is a domain independent ontology that contains domain independent concepts and predicates classes about exceptional conditions. It identifies, summarizes and defines different categories of exceptions types that might happen during the execution of the SPL-based business process. This ontology can be used in different domains. In order to deal with the exceptions related to the particular domain, the concepts in the ontology can be augmented using domain concepts to provide the exception handling agents with a capability of processing the exceptions in relation to domain level concepts. The context ontology contains any concepts that are relevant to the interactions between a user and an environment. It includes knowledge about the facts, constraints and the assigned tasks, which is owned and maintained by agents.

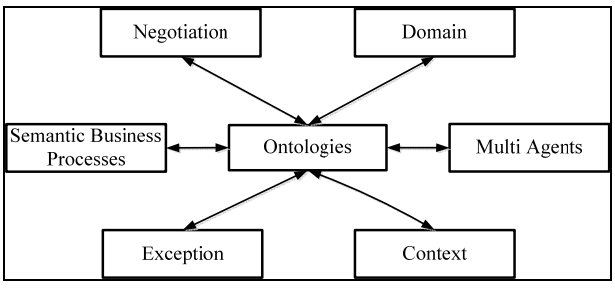

Figure 6. Ontology-centered method.

It is depending on the exception occurred, agents with the desire to handle exceptions would use context ontology. Our framework includes five kinds of ontologies:

- Domain ontology: it represents a conceptualization of the specific domain the framework will be applied, so it contains knowledge items (concepts, attributes, relationships and axioms etc.) that model the application in which the framework is to be used, and supports the communication among the component agents in the framework without misinterpretations.
- Context Ontology: it contains knowledge about the environment for each agent possesses. It generally includes knowledge about the facts, constraints and the assigned tasks, which is owned and maintained by agents.
- Exception Ontology: it identifies, summarizes and defines different categories of exceptions types that might happen during the execution of the SPL program. It is important to note that described exception types are derived only from the logic of the SPL program and they are neutral to the purpose for which they can be used, exception ontology is application independent.
- Negotiation ontology: it contains knowledge about the negotiation mechanisms for component agents coordinating their interactions.
- Semantic Web Services Ontology: this ontology is the main part of ontologies libraries. It contains web service functional capabilities, Non-functional properties, mediation information, knowledge for supporting discovery and orchestration, and grounding information.

In summary, the framework is independent both of the specific domain and specific practical application. It can be used as a reference and implemented in a complex business environment. Ontologies provide semantic links between the knowledge level components describing semantic web services and the conditions related to its use, which also part of the RDF4S.

IV. RELATED WORK

A. Semantic Web Services and Agents

The implementation of frameworks or infrastructures is one of the major challenges towards the description, reasoning and execution of Semantic Web services. Different integrated frameworks have been presented. OWL-S describes services in OWL-S ontology and use OWL reasoning capabilities to determine whether the features of the services to be composed match with the functionality the new service requires. OVM [11] is a generic implementation of OWL-S. ODE SWS [12] considers a PSM-based approach that enables semantic Web services design and composition in knowledge level with a language-independent manner. WSMO provides goal-based framework to discovery, composite, and invokes semantic Web services. It has two reference implementations: IRS-III [13] and WSMX [14]. Meteor-S [15] introduces framework for configuring and executing

dynamic Web processes that focus on flexible process composition and QoS properties. On the other hand, numerous research projects have been carried out that try to put semantic Web services and agents technologies together into integrated frameworks. However, there is much discussion on how to actually achieve this integration (e.g. the Semantic Web FRED project [16], Goal-Oriented Discovery for SWS system [17]). These solutions provide many different methods of combining agent technology with ontologies and semantic Web services in order to develop a system for automated cooperation. The main difference between our approach and those proposals mentioned above is that we introduce SPL to describe business logic and design a set of lightweight-agents for building up the run-time infrastructure which can quickly respond to competitions and changing regulations. These approaches cannot handle the issue of runtime adaptability and dynamism.

B. Exception Handling and Web Service-based Business Process Management

We review some efforts on exception handling in workflows, especially in business process based on Web services. Casati et al. [18] design an active rule-based language to specify fault-handling logic in workflow systems. Liu et al [19] define Event-Condition-Action (ECA) exception handling rules combined with the WS-BPEL process model. Similarly, Zeng et al. [20] propose a policy driven exception management framework for composite Web services. All these works have same idea of separating normal control flow from exception flow. Furthermore, Cao et al [21] introduce an approach to handling exceptions based on mobile agent in distributed Workflow Management System. When dealing with exceptions in service-based business processes, Wang et al. [22] research the issue of dynamic and fault tolerance Web services composition based on WS-BPEL fault handling mechanism and development a constraint integration and violation handling technique. Friedirich et al [23] describe a model-based approach to handle exceptions in service-based processes and to repair the faulty activities based on self-healing and planning techniques. Once an exception arise, the support tool platform can monitor, diagnosis, generate and execute corresponding repair plans. Regarding exception handling of semantic Web services, Vaculin et al [24] design an method for specification of exception handling and recovery of semantic Web services based on OWL-S. They extend the OWL-S process model definition with fault handlers, constraint violation handlers and event handlers, and introduce explicit recovery actions for recovery from errorsand violations of constraints. Moller and Schuldt [25] propose an approach to flexible and automatic semantic failure handling for composite services. Our approach shares many similarities with their work and augments to facilitate the reason of semantically equivalent services. By Comparing with these works, our approach support directly to define exception handling logic in the process model of SPL program, and proposed framework will provide flexible forward recovery for reliable execution of SPL program.

V. CONCLUSION AND FUTURE WORK

The paper introduces an ontology based exception handling method, which integrating semantic Web services, agents and ontologies for Semantic Business Process using SPL. The proposed framework aims to exploit the striking potential of these technologies based on SPL and support for reliable executing business process based Semantic Web Services, the MAS can change the behavior of business processes and handle exceptions without redeployment. While SPL and the framework cover some adaptability and dynamism of service-oriented application, there are other adaptive and dynamic aspects that are not currently covered which including transaction and self-healing mechanism. We are currently working to address these aspects.

ACKNOWLEDGMENT

This work was supported in part by a grant from the National High Technology Research and Development Program of China (863 Program) (No. 2006AA01Z168), National Natural Science Foundation of China (No. 61070012/F020202), Xinjiang Uygur Autonomous University Research Fund for Young Teacher (NO. XJEDU2009S15) and Xinjiang University Doctoral Graduate Research Startup Fund (NO. BS090142).

REFERENCES

[1] M. Burstein, C. Bussler, M. Zaremba, T. Finin, M. N. Huhns, M. Paolucci, A. P. Sheth, S. Williams and M. Zaremba, “A Semantic Web Services Architecture,” IEEE Internet Computing, vol. 19, no. 5, Sept.-Oct. 2005, pp. 72-81, doi: 10.1109/MIC.2005.96.

[2] M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. Fen sel. “Semantic Business Process Management: a Vision towards Using Semantic Web Services for Business Process Management,” Proc. IEEE International Conference on e-Business Engineering (ICEBE 2005), IEEE Press, Oct, 2005, pp. 535-540, doi: 10.1109/ICEBE .2005.110.

[3] BPEL. Web Service Business Process Execution Language Version 2.0. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf, April 2007.

[4] OWL-S. Semantic Markup for Web Services. http://www.w3.org/submission/OWL-S/, 2004.

[5] D. Roman, H. Laursen and U. Keller eds, “Web Service Modeling Ontology-standard,” WSMO working draft, http://www.wsmo.org/2004/d2/v0.2/20040306.

[6] J. Opecky, T. Vitvar, C. Bournez,and J. Farrell. “SAWSDL: Semantic Annotations for WSDL and XML Schema,” IEEE Internet Computing, vol. 11, no. 6, Nov. 2007, pp. 60-67, doi: 10.1109/MIC.2007.134.

[7] J. Nitzsche, T.van Lessen, D. Karastoyanova, and F. Leymann. “BPEL for Semantic Web Services (BPEL4SWS),” On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, Springer, LNCS, vol. 4805, pp.179-188, doi:10.1007/978-3-540-76888-3_37.

[8] H. Cao, S.Ying, D. Du, and Y. Xiao. “Orchestrating Semantic Web Services with Semantic Programming Language,” Proc. IEEE International Workshop on Semantic Computing and Systems (WSCS 2008), IEEE Press, July, 2008, pp. 101-106,doi:10.1109/WSCS.2008.22.

[9] P. A. Buhler, J. M. Vidal, “Towards adaptive workflow enactment using multiagent systems,” Information Technology and Management, vol. 6, no. 1, Jan. 2005, pp. 61-87, doi: 10.1007/s10799-004-7775-2.
[10] D. Xie, S. Ying, J. Yao, B. Xiao, “A Resource Description Framework for Service,” Proc. International Conference on Wireless Communications, Networking and Mobile computing (WiCom 2007), IEEE Press, Sept. 2007, pp. 3351-3354, doi: 10.1109/WICOM.2007.830.
[11] M. Paolucci, A. Ankolekar, N. Srinivasan, and K. P. Sycara. “The DAML-S virtual machine,” Proc. International Semantic Web Conference (ISWC 2003), Springer-Verlag, Oct. 2003, vol. 2870, pp. 290–305, doi: 10.1007/978-3-540-39718-2_19.
[12] A. Gomez-Perez, R. Gonzalez-Cabero, M. Lama, “ODE SWS: a framework for designing and composing semantic Web services,” IEEE Intelligent Systems, vol. 19, no. 4, May. 2005, pp. 24-31, doi: 10.1109/MIS.2004.32.
[13] F. Hakimpour, D. Sell, L. Cabral, J. Domingue, and E. Motta. "Semantic Web service composition in IRS-III: the structured approach," Proc. Seventh IEEE International Conference on E-Commerce Technology (CEC 2005), IEEE Press, July, 2005, pp.484-487, doi: 10.1109/ICECT .2005.79.
[14] A. Haller, E.Cimpian, A. Mocan, E. Oren, C. Bussler. "WSMX - a semantic service-oriented architecture," Proc. IEEE International Conference on Web Services (ICWS 2005), IEEE Press, July 2005, pp. 321- 328, doi: 10.1109/ICWS.2005.139.
[15] R. Aggarwal, V. Kunal, J.Miller, and W.Milnor. "Constraint driven Web service composition in METEOR-S," Proc. IEEE International Conference on Services Computing (SCC 2004), IEEE Press, Sept. 2004, pp. 23-30, doi: 10.1109/SCC.2004.1357986.
[16] M. Stollberg, D. Roman, I. Toma, U. Keller, R. Herzog, P. Zugmann and D. Fensel. "Semantic Web Fred - Automated Goal Resolution on the Semantic Web," Proc. of the 38th Annual Hawaii International Conference on System Sciences (HICSS '05), IEEE Press, Jan. 2005, pp. 111c, doi: 10.1109/HICSS.2005.536.
[17] J. M. Gomez, M. Rico-Almodovar, F. Garcıa-Sanchez, I. Toma, and S. Han. “GODO: Goal oriented discovery for semantic webservices,” Discovery on the WWW Workshop (SDISCO’06), 2006, Beijin, China.
[18] F. Casati, S. Ceri, S. Paraboschi, and G. Pozzi, “Specification and Implementation of Exceptions in Workflow Management Systems,” ACM Transactions on Database Systems, vol.24, no.3, Sept. 1999, pp.405-451, doi: 10.1145/328939.328996.
[19] A. Liu, Q. Li, L. Huang, M. Xiao. "FACTS: A Framework for Fault-Tolerant Composition of Transactional Web Services." IEEE Transactions on Services Computing, vol.3, no.1, Jan.-March 2010, pp.46-59, doi: 10.1109/TSC.2009.28.
[20] L. Zeng, H. Lei, J. Jeng, J. Chung, and B. Benatallah, “Policy-Driven Exception Management for Composite Web Services,” Proc. Seventh IEEE International Conference on E-Commerce Technology (CEC 2005), IEEE Press, July 2005, pp. 355-363, doi: 10.1109/ICECT.2005.66
[21] J. Cao, J. Yang, W.T. Chan, and C. Xu, “Exception Handling in Distributed Workflow Systems Using Mobile Agents,” Proc. IEEE International Conference on e-Business Engineering (ICEBE 2005), IEEE Press, Oct. 2005, pp. 48-55, doi: 10.1109/ICEBE.2005.65.
[22] M. Wang, K.Y. Bandara, C. Pahl. "Constraint Integration and Violation Handling for BPEL Processes," Proc. Fourth International Conference on Internet and Web Applications and Services (ICIW 2009), IEEE Press, May 2009, pp.337-342, doi: 10.1109/ICIW.2009.56.
[23] G. Friedrich, M. Fugini, E. Mussi, B. Pernici and G. Tagni. "Exception Handling for Repair in Service-Based Processes," IEEE Transactions on Software Engineering, vol. 36, no. 2, March/April 2010, pp: 198-215, doi: 10.1109/TSE.2010.8.
[24] R. Vaculin, K. Wiesner, K. Sycara. “Exception Handling and Recovery of Semantic Web Services”. Proc. Fourth International Conference on Networking and Services (ICNS 2008). IEEE Press, March 2008, pp. 217-222, doi: 10.1109/ICNS.2008.35.
[25] T. Moller and H. Schuldt. “OSIRIS Next: Flexible Semantic Failure Handling for Composite Web Service” Proc. IEEE Fourth International Conference on Execution. Semantic Computing (ICSC 2010), IEEE Press, Sept. 2010, pp. 212-217, doi: 10.1109/ICSC.2010.38.

Kai Zhao is a Ph.D. candidate of State Key Lab of Software Engineering (SKLSE) at Wuhan University (WHU) in Wuhan, China. He received his M.S. degree in Computer Science from Dalian University. His research interests include the Semantic Web, Web Service and Service-Oriented Architecture.

Linlin Zhang received the Ph.D. degree in Software Engineering from State Key Lab of Software Engineering (SKLSE) at Wuhan University (WHU), China, in 2009. She is currently an associate professor at School of Information Science and Engineering, Xinjiang University. Her research interests include trustworthy software, Aspect-Oriented software engineering and Service-Oriented architecture.

Shi Ying received the Ph.D. degree in computer science from the Wuhan University, China, in 1999. He is currently a professor at Wuhan University, where he is vice dean in the Computer School and he is also the deputy director of the State Key Laboratory of Software Engineering. He has authored or coauthored more than 100 referred papers in the area of software engineering. His current research interests include service-oriented software engineering, Semantic Web service, and trustworthy software.