ISSN:0975-9646

Cross Domain Composition of Web Service Workflows using a Provenance Ontology with an automated Re-planning

B.Meenakshi Sundaram
Asst. Proffessor
Department of Computer Applications
School of Computer Science and Technology
Karunya University, Coimbatore, India.

Dr. D. Manimegalai
Prof & Head
Department of Information Technology
National College of Engineering
Kovilpatti, India.

Abstract-Workflow is a sequence of processes through which a work request completes from inception till end by using multiple service providers across domain boundaries. Workflow Web services are interoperable and provides an interface for remote clients to get information or to get workflow templates. A domain specific and domain independent OWL ontology (PROV) to represent workflow specifications to trace workflow execution is developed. To complete a business transaction requests, there arise a need of combining multiple workflow web services to form a composition with the help of multi-agent system. In this paper, a conceptual model of provenance ontology (PROV) for a workflow with the help of multihoming agent based fault tolerant cross domain composition system is proposed. A few recently proposed Composing web services enacted by autonomous agents through agent-centric contract net protocol, Web Service Composition using Provenance and Automatic service composition using Partially Observable Markov Decision Processes and provenance have been studied thoroughly. This paper presents a contemporary way to automatically compose web service workflow that uses active transactional component web services. The web services workflows are described with open provenance workflow ontology (OPWO). The PROV-O can be used to describe both component and web service workflows. In addition it also describes the dependent service flow to ease the automated process. The Provenance ontology gives workflow execution traces as well as more abstract reusable workflows. Workflow transactions can also use provenance information to understand user's query. We have also implemented a workflow engine that runs all possible workflow use-cases. Here we analyze using our ontology and some workflow instances with a reasoning agent to automatically compose the workflow that fulfills given requirements. The outcome from the derived and combined workflow instances can be executed using our workflow engine.

Keywords: Provenance, Web service workflow composition, Multihoming Agent fault tolerant composition etc.

1. INTRODUCTION

The Business Artifact is a digital representation of a business model. Artifact-centric Business Process Model is comprised of process, data and entities. A web service is the business process components communicate through common messaging structures which is strength of interoperability of distributed component systems. The proposals like BPEL4WS, XLANG and WSFL have been presented for describing workflows of composed web services in relatively stable environments. We propose an automatic workflow composer with re-planning facility.

Web services workflow is a well-defined set of abstract process definition to each process whose output depends on other previous web services. Composed workflows combine web services from several individual web services to one combined workflow, which may not show all internal operations. Because of most of the business logics execution are done in background. Composed web service workflow includes dependencies between operations inside the web service and between component web services. Transactional workflow guarantees that all services in the workflow either suspend or fail. Concurrent and long-term transactions must be handled without interruptions. Currently, most of the web service workflows are developed and maintained manually. [4]

The Semantic Web is a Web as a whole can be made more intelligent and perhaps even intuitive about how to serve a user's needs. [1] OWL-S enables to create the ontology for a domain and the instantiation of these ontologies in the description of web service specification. In this paper, we are interested in automated reasoning.

Generally the composition of web service workflows needs a reasoning engine (planner), a set of inferences, and semantic information about the services and workflows. Provenance ontology (PTWO) enable automated semantic reasoner.

With dynamic automated composing of web services, there are more dependencies between component services. We describe here a software agent that does the workflow planning based on the semantic requirement information. The agent offers a single interface to multiple services. Changes in component services cause a re-planning.

2. RELATED WORK

Most research concerning web service workflows has been about manual or semi-automated composition of services to provide higher-level services [2].

Automatic composition of web services using situation calculus, pi-calculus, and process algebra has been presented in the earlier researches [3]. They have transformed a significant portion of semantics to first-order logic and also to Petri nets. Petri nets have been used to simulate and verify the workflows. Their work proves that translation of workflows to Petri nets is possible, and that

meaningful deduction can be made. We extend their research by combining separate workflow usecases using a semantic agent to one automatically generated workflow.

Self-Serv [4] is a peer-to-peer (P2P) approach to web service workflow composition. The overall workflow is managed without a central planner or coordinator. Instead, Self-Serv has coordinating messages between state coordinators. SWORD [5] is a rule-based developer toolkit for web service composition. It generates a plan based on input/output conditions specified for all component services. SWORD uses its own web service platform. SCET [9] is a tool for automatic off-line composition and execution of web service workflows. It generates Perl execution code for WSFL (Web Services Flow Language from IBM) workflow specifications.

The significance of provenance information in SOA has been progressively perceived and various methodologies have been created to give a provenance system to catch such information. Tsai et al. [14], [15] provide a dynamic framework for classification and collection of provenance data in SOA systems, their emphasis is fundamentally on security, unwavering quality and honesty of information directed through a SOA framework instead of the structure of the provenance information gathered. Michlmayr et al. [16] present an approach for capturing service runtime events, but their work again focuses on security issues such as data integrity and access control mechanisms as its foundation. Rajbhandari et al. [17] propose an approach for recording provenance in SOAs, including a scalability analysis of the effect of increases in the provenance data collection. Their provenance model for Web service architectures focuses on capturing the provenance data and representing them in a standard format, and on querying and reasoning over the provenance of process instances.

3. ONTOLOGY

By ontology means a conceptual specialization and their relations in a machine-understandable way. The Provenance ontology to provide the foundation to implement provenance applications in different domains that can represent, exchange, and integrate provenance information generated in different systems and under different contexts.

3.1 PROV ontology

PROV-O [11]] accommodates all different uses of provenance. Distinct individuals may have alternate points of view on provenance, and subsequently diverse sorts of data may be caught in provenance records. One viewpoint (prov: Agent) may concentrate on agent focused provenance, that is, the thing that individuals or associations were included in creating or controlling the data being referred to. A second viewpoint (prov: Entity) may concentrate on object focused provenance, by following the starting points of bits of a report to different archives. A third viewpoint (prov: Activity) one may take is on process focused provenance, catching the moves and steps made to produce the data being referred to. Since our work is based on workflow, we consider the third viewpoint of PROV-O to compose the process workflow.

Table 1. PROV-O Task types

Task type	Description
prov:startedAtTime	when an activity is deemed to have started
prov:used	start of using an entity by an activity
prov:wasGeneratedBy	Completion of production of a new entity by an activity.
prov:wasInformedBy	exchange of an entity by two activities
prov:wasDerivedFrom	derivation is a transformation of an entity into another
prov:endedAtTime	when a activity is considered to have finished

3.2 P-Plan

P-plan[11] is a PROV extension helps to specify the planned execution activities. OPMW extends PROV, OPM and P-Plan to get the process view provenance.

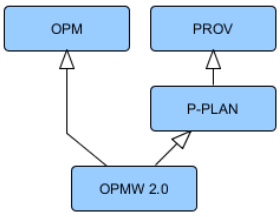


Fig. 1. OPMW as an extension of OPM, PROV and P-plan [11]

Table 2. P-Plan Task types

Task type	Description
p-plan:Steps	Represents the planned execution activity.
p-plan:correspondsToStep	links a p-plan:Activity to its planned p-plan:Step
p-plan:isPrecededBy	links a p-plan:Step to the p-plan:Step preceeding it.
p-plan:isStepOfPlan	links a p-plan:Step to the p-plan:Plan
p-plan:isVariableOfPlan	Variable to the p-plan:Plan

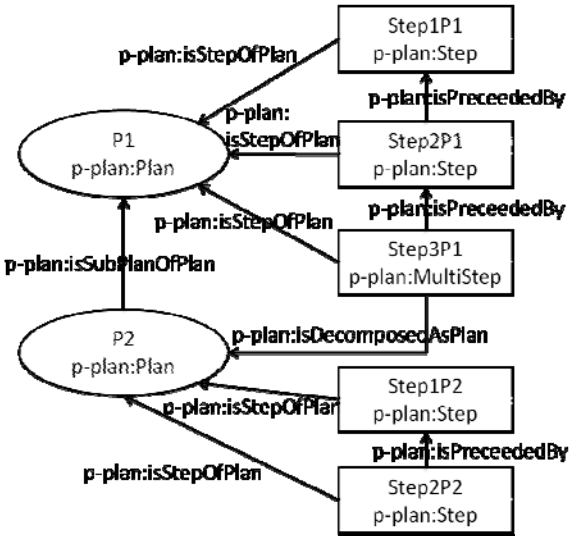
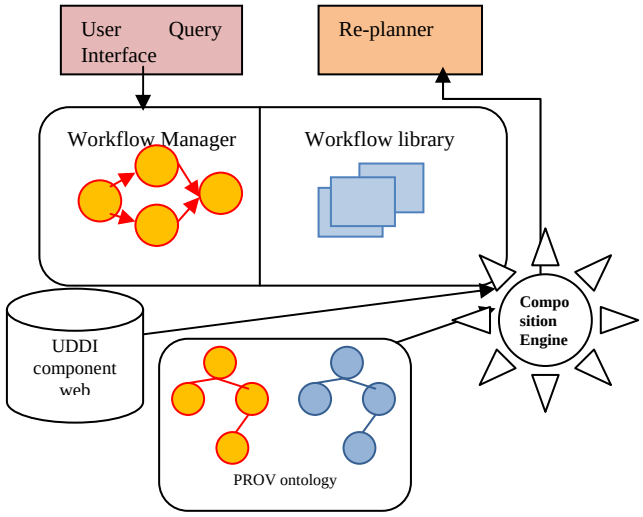


Fig 2: Sub-plan representation in P-PLAN [11]

The workflow experts draw the p-Plan activities and a sub-plan to compose the web services.

4. PROPOSED WORK

The workflow web services are identified for a We have implemented a workflow engine that reads and executes a Transactional ontology-based web service workflow instance.



Error! Not a valid bookmark self-reference.3. presents the overall system architecture.

The core modules are Workflow Manager and Workflow library, Transact Ontology, PROV Ontology. Workflow Manager is responsible for loading, creating, saving and retrieving component workflow instances. Transact Ontology module implements reading, writing, updating and validating a workflow instance. Workflow Engine module holds the workflow execution logic. WS Planner module uses a workflow instance to send messages to web services as instructed by the workflow engine.

The implementation uses Java Web Service Developer Pack [13] for web services. Protégé with Jena for ontology reading and writing.

In our experience, the workflow instance specifications using PROV ontology offers the following benefits:

- Tasks and other concepts can be kept in an order.
- Inferences are written in the ontology, instead of being in parser code.
- Integration of workflow ontology concepts and PROV ontology for composition.
- Class-based specification is easy to map to implementation classes.

Our main motivation for developing an ontology-based workflow engine was to enable sharing and combining workflows using automated re-planning facility.

5. PLANNING AGENT

Planner's role is to plan a workflow that accomplishes the user needs. The workflow is created at run-time, based on following:

- Workflow library information about web services
- Workflow instances for component web services
- Starting/Ending Workflow instance
- User input parameters
- Inferences
- Workflow composition

Workflow instances are described using our Transactional ontology and Provenance ontology. Using ontology enables us to also reason about the transaction models. We use Provenance profile ontology to describe the common meaning of services.

We have proposed [8] a separate *Workflow Manager* that stores explicitly registered workflow instances in the same sense as UDDI registry stores web service interface descriptions. Workflow Manager enables planning agent to select services based on their workflow and transaction models. The planning agent can automatically make the composition based on the stored workflows.

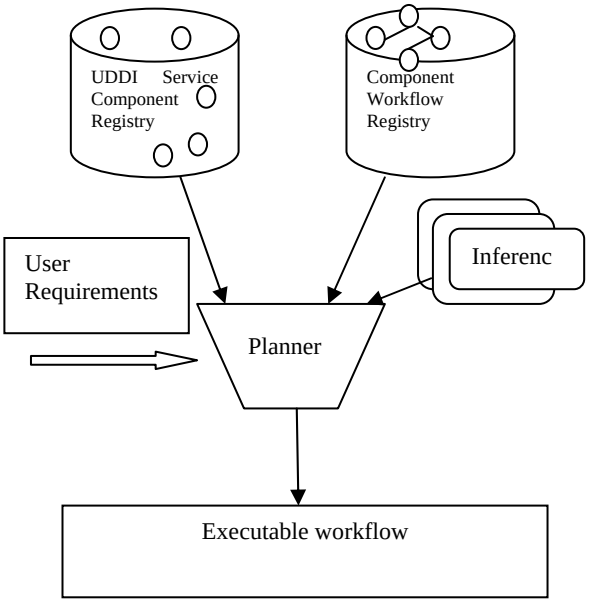


Fig. 4. Planning agent input and output

Figure 4 presents the inputs and output of the planning agent.

6. COMPOSITION

Four workflow specification components are needed to be able to compose the workflow [10].

- Tasks in sequence
- Transactional requirements in order
- Directed Data flow
- Executable composition structure

The tasks specify what component services are used in what sequence. Modular design of tasks helps reuse them in composed master workflow. Transactional requirements in order are described for all web services in our ontology. Data flow and executable structure are objectives of the composition.

Executable structure (composite workflow) is composed from the master workflow and the component workflows. First, Transaction component Service Registry is used to find the correct services. Second, their workflows are read from the Provenace Workflow Registry. Third, ordering constraints in the master workflow are added to the

knowledge base. Fourth, some tasks are added to the workflow. After this step, workflow has all tasks and control links. Finally, data links may be added.

Order Tasks will create a partial ordering for the tasks in component workflows. It is not necessary to order all tasks separately, as the component workflows already have ordered some tasks. Data dependencies specified in the master workflow need to be used in planning as well.

Add Tasks phase adds Fork/Join control links around sets of concurrent tasks. If the master workflow specifies that component workflows are executed in sequence, a Go control link is added between their tasks.

Our workflow engine implementation supports named data links between tasks. By default, all data links are internal to component workflows only. Master workflow can specify some data links to be public and usable by other component workflows. In practice data links often require filters or transformations. [10] There is some research on ontology version control and ontology transformations that could be useful.

7. CONCLUSIONS

We have implemented a web service workflow composition engine that uses Provenance ontology languages to select and execute workflows. This paper presents a way to use provenance ontology-based reasoning to automatically combine component workflow instances. Provenance helps in many ways than typical ontology process model, which lets us to use inference engines. In future, we plan to develop the composition framework for cross enterprise workflow integration.

REFERENCES

[1] Tim Berners-Lee, James Hendler, and Ora Lassila, “The semantic web”, *Scientific American*, May 2001.

[2] Joe Kopena and William C.Regli, “DAMLJessKB: A Tool for Reasoning with the Semantic Web”. http://edge.mcs.drexel.edu/assemblies/software/damljesskb/articles/DAMLJessKB-2002.pdf

[3] Narayanan, S. and McIlraith, S., "Simulation, Verification and Automated Composition of Web Services", *Proceedings of the Eleventh International World Wide Web Conference*, 2002.

[4] Boualem Benatallah. and Marlon Dumas, "The Self-Serv Environment for Web Services Composition", *IEEE Internet Computing*, Jan-Feb, 2003.

[5] Lasse Pajunen, Jarmo Korhonen, Juha Puustjärvi. “Adaptive Web Transactions: An Approach for Achieving the Atomicity of Composed Web Services”, Proceedings of EuroWeb conference, 2002.

[6] HP Labs, Jena Semantic Web toolkit. http://www.hpl.hp.com/semweb/jena.htm

[7] Shankar R. P. and Armando F. “SWORD: A Developer Toolkit for Web Service Composition”, *Proceedings of the Eleventh International World Wide Web Conference*, 2002. http://www2002.org/CDROM/alternate/786/

[8] Jarmo Korhonen, Lasse Pajunen, Juha Puustjärvi, “Using Transactional Workflow Ontology in Agent Cooperation”, AIM *Workshop, First EurAsian Conference on Advances in ICT*, 2002. http://www.iki.fi/jako/papers/twfo.pdf

[9] Ruoyan, Z., Arpinar, B., Aleman-Meza, B., Automatic Composition of Semantic Web Services, The 2003 International Conference on Web Services (ICWS'03), June 2003. http://lsdis.cs.uga.edu/lib/download/composition_short_icws.doc

[10] Juha Puustjärvi, Henri Tirri, Jari Veijalainen. “Reusability and modularity in transactional workflows”. *Information Systems*, Vol 22, No 2/3, pp 101-120, 1997

[11] DAML-S:Semantic Markup for Web Services. P*roceedings of the International Semantic Web Working Symposium (SWWS)*, 2001. http://www.daml.org/services/SWWS.pdf

[12] BPEL4WS–Business process execution language for web services, http://www.ibm.com/developerworks/webservices/library/ws-bpel/

[13] JWSDP – Java Web Services Developer Pack. http://java.sun.com/webservices/webservicespack.html

[14] T. Wei-Tek, W. Xiao, Z. Dawei, P. Ray, C. Yinong, and C. Jen-Yao, “A New SOA Data-Provenance Framework,” in Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems, ser. ISADS ’07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 105–112. [Online]. Available: http://dx.doi.org/10.1109/ISADS.2007.5

[15] T. Wei-Tek, W. Xiao, C. Yinong, P. A. Raymond, C. Jen-Yao, and Z. Dawei, “Data provenance in SOA: security, reliability, and integrity,” Service Oriented Computing and Applications, vol. 1, no. 4, pp. 223– 247, 2007.

[16] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar, “Service Prov enance in QoS-Aware Web Service Runtimes,” in Proceedings of the 2009 IEEE International Conference on Web Services, [Online]. Available: http://dx.doi.org/10.1109/ICWS.2009.32

[17] S. Rajbhandari and D. Walker, “Incorporating Provenance in Service Oriented Architecture,” in Proceedings of the International Conference on Next Generation Web Services Practices, ser. NWESP ’06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 33–40. [Online]. Available: http://dx.doi.org/10.1109/NWESP.2006.18