Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2006 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2006

Dependency Management in Web Services Composition

Jong Woo
Georgia State University

Jordi Conesa
Universitat Politecnica de Catalunya

Radhika Jain
The University of Memphis

Dependency Management in Web Services Composition

Jong Woo Kim
Department of Computer Information Systems
Georgia State University
Atlanta, GA
jkim@cis.gsu.edu

Jordi Conesa
Departament de llenguatges i sistemas informàtics
Universitat Politecnica de Catalunya
Jordi Girona 1-3, 08034 Barcelona
jconesa@lsi.upc.edu

Radhika Jain
Department of Management Information Systems
The University of Memphis
Memphis, TN
r.jain@memphis.edu

ABSTRACT

Dependencies among the tasks performed by web services affect how the web services are composed. We propose an approach to managing dependencies among tasks in order to support web services composition. We define BT-DML, a markup language to express dependencies. We describe a prototype system called web services manager (WSM) that implements this language to support various tasks in web services composition.

Keywords

Web service, composition, dependencies, DML

INTRODUCTION

Service Oriented Architecture (SOA) is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired results for a consumer. Both provider and consumer roles are played by the software agents on behalf of their owners (Hashimi, 2003). Modularized, self-describing nature of web services makes them a very promising technology for realizing SOA. As web services become the basic building blocks from which new applications are created, service composition is a major concern in the application development process (Curbera, Khalaf, Mukhi, Tai and Weerawarana, 2003). To achieve a complex service-based application in a SOA environment, orchestration and composition of web services is crucial.

Literature in the area of web services has focused on addressing various issues such as quality of service, semantic matchmaking, and web services composition. Milanovic and Malek (2004) distinguish between the composition and orchestration of web services. According to them, orchestration includes only functional properties while composition includes non-functional properties as well. Standards such as WS-BPEL are primarily concerned with the orchestration of web services (i.e. addressing the connectivity issues). In this paper, we are focused on supporting web services composition.

Recent research on web services composition has proposed several interesting solutions. While point-to-point simple composition can be done easily, achieving complex configurations is very difficult. Colman and Han (Colman and Han, 2005a) propose a very novel approach which tries to include both behavioral and non-functional factors in creating compositions. They suggest creating an abstract management and coordination layer for web services composition. Such a layer helps encapsulate functions as services accessible through a WSDL interface. Our research is based on the premise that the implementation of this layer using a middleware-oriented approach for composition will be most effective only if the developers have access to knowledge about the context in which web services composition is achieved. Specifically, when developers understand the rationale behind compositions, they can develop more adaptive and higher quality system quickly. Therefore, we propose a separate management layer that focuses on the management of dependencies among the tasks supported by web services. Extending the approach proposed by prior research (Kim and Jain, 2005), we propose a language to model these dependencies using rules. We also develop a support system that implements this language to support various tasks in web services composition.

In a SOA environment, applications will be comprised of a number of modular web services. Composing web services to satisfy business needs of an organization in a dynamic environment is a challenging task. Managing dependencies is an essential activity in improving system quality (Ramesh and Jarke, 2001). As web services emerge as a new paradigm for the software development, better composition of modular web services-based applications can be facilitated by the ability to manage dependencies among tasks supported by the services. While the current literature on the web services composition addresses various aspects such as automated, quality-driven, and semantic composition, they do not adequately address the need to manage the dependencies among tasks supported by web services. We argue that effective management of these dependencies is critical for achieving dynamic composition of web services.

In this article, we present a task dependency based approach for web services composition. Rules are used to represent the dependencies among tasks enabled by web services. Our work addresses two primary research questions: 1) How can the dependencies among tasks supported by web services be represented, and 2) how can web services composition be supported by tools to manage these dependencies.

In the following sections, we briefly present related work on web services composition, workflow dependencies and rules. We then present our conceptual framework for web services composition based on task dependencies. This discussion is followed by a description of BT-DML, an XML based markup language that we have defined to represent dependencies. Then we present WSM, our prototype system that implements this language. We use a scenario to illustrate our approach and the functionalities of WSM.

RELATED RESEARCH

This section reviews related literature in the areas of web services composition and workflow dependencies to highlight the motivation for our research and develop our research approach.

Workflow dependency

The literature in the area of workflow specifications and execution suggests several approaches to the management of dependencies. For example, Choi et al (2002) introduce the concept of task states and state dependencies to allow the user to express finer relationships between tasks than basic workflow specifications (Choi, Park and Lee, 2002). Dependencies among task states can be classified into passive and active dependencies. Task state dependencies are further divided into intra-task dependencies which exist among states of the same task and inter-task dependencies which exist among states of different tasks. Similarly, Senkul and Toroslu (2004) specify a scheduling architecture for a workflow execution under resource allocation constraints as well as commonly addressed temporal and causality constraints that result in existence and order dependencies. Schulz and Orlowska (2004)distinguish between outsourced workflows and distributed workflows. They suggest using state dependencies for outsourced workflows and control-flow dependencies for distributed workflows. With control flow dependency, communication occurs through synchronization of tasks and dependencies. Control flow dependencies provide a loose coupling between workflows, because they merely pass state and workflow-relevant data from one workflow to another.

This literature stream is specifically concerned about the representation of dependencies to support the automation of workflow execution. At the execution level, the primary concern for managing dependencies is the efficient and effective routing of workflow items when changes are necessary. But, workflow automation and execution assumes the presence of well-defined series of tasks at a higher (abstract) level. In fact, the focus of our work is to support the specification and management of dependencies at this level (i.e, among tasks). In contrast to prior work that is primarily concerned with supporting the execution level, our work is focused on supporting the conceptual design which hasn't received much attention in the current literature. Our work is similar in spirit to that of Dellen et al. (1997) who suggest that dependencies can be effectively used to enhance the flexibility of coordination in workflow management systems (Dellen, Maurer and Pews, 1997). In this paper, we define task dependencies in the context of web services and illustrate how they may be used to support conceptual design tasks.

Web services composition

Web services standards organizations are working on various composition standards. However, several current and proposed standards such as WS-BPEL, XLANG, and WSFL (Medjahed, Bouguettaya and Elmagarmid, 2003) do not address conformance and quality of service issues. Thery are not well suited to many of the automated reasoning tasks such as the one addressed in our research.and envisioned by semantic web services (Aral, 2004; McIlraith, Son and Zeng, 2001) OWL-S, Ontology-based semantic approach for the composition of web services is difficult to learn and has an imprecise underlying conceptual model leading to multiple modeling possibilities and parametric polymorphism (Sabou, Richards and Spluter,

2003).

Many of web services composition approaches focus primarily on finding appropriate web services for carrying out tasks and automating their composition (Curbera et al., 2003; McIlraith and Mandell, 2002; Peterson, 2003; Zeng, Benatallah and Dumas, 2003). Such approaches are appropriate as long as the existing composition is unchanged because they assume the presence of predefined steps for tasks and activities. Mandell and Mcllraith (2003) acknowledge that automated web services composition requires a fundamental shift in industrial frameworks from executing predefined process models to computing and adapting execution plans from evolving objectives.

Several ongoing efforts extend composition standards in a variety of ways. For example, while BEA and W3C are concerned with improving web services coordination, service selection has been an agenda for OASIS and UDDI. Also, WSRF and WS-Policy also represent this trend. Cremona framework based on WS-Agreement (Ludwig, Dan and Kearney, 2004) proposes an organizational approach for creating adaptive system by focusing on external contracts. Finally Aspect-oriented approaches also attempt at improving web services composition (Baligand and Montfort, 2004; Colman and Han, 2005b). Unlike WS-BPEL, a de-facto standard of web services composition, our approach is not a means of developing executable compositions. Rather, it focuses on helping developers' design decision making processes during web services composition.

A composition of web services may need to change for a variety of reasons, say for example, due to changes in the market conditions or changes in the customer requirements. Such changes in the environmental conditions are usually reflected in the business rules of the organization and require modifications to the existing composition of web services. Such re-composition is guided by various dependencies among the affected tasks (Malone, Crowston, Lee, Pentland, Dellarocas, Wyner, Quimby, Osborn and Bernstein, 1999). Without appropriate tools for managing these dependencies, the ability to understand the rationale for the current composition and making modifications to it are very challenging (Ramesh and Jarke, 2001). This task requires improved means to describe dependencies to enable the capture of the rationale behind various decisions made during composition. Our research seeks to develop an approach to address this problem which is described in the next section.

OUR APPROACH TO MANAGING TASK AND BUSINESS RULES DEPENDENCIES

We propose an approach for managing dependencies among tasks using rules to support the following critical activities in the composition of web services:

- Represent the rationale behind a composition from a business perspective
- Track the composition and hierarchy of tasks and related rules
- Manage the repercussions of changes to compositions necessitated by changes in the environment

Malone et al. suggest different types of dependencies among the tasks and the resources consumed by these tasks, namely flow, sharing, and fit dependency (Malone et al., 1999). Flow dependency suggests that the output of one task is the input of another task and is present implicitly in sharing and fit dependencies. Sharing dependency suggests that the same resource is used by the multiple tasks or activities. Finally, fit dependency suggests that multiple activities collectively produce one resource or output. Further, the dependency types proposed by Yu and Mylopoulos provide various semantics to dependencies (Yu and Mylopoulos, 1994). They classify dependencies based on the ontological categories of the dependum (the object around which two actors have dependency), namely goal, resource, and task dependencies. In goal dependency, a depender depends upon a dependee to make a condition in the world come true. In task dependency, a depender depends upon a dependee to perform some activity, while the goal for having performed an activity is not given. Finally, in resource dependency a depender depends upon a dependee for the availability of an entity. They additionally distinguish among the degrees of strength of a dependency as open, committed, or critical. These dependency types provide an overall framework for our task dependency model.

Prior research (Kim and Jain, 2005) identifies three types of dependencies that apply to web services, namely *interface, quality of service (QoS), and protocol mediation* dependency. In interface dependency, since no human is involved, tasks to be accomplished by the involved web services need to be well-defined and their input and output interfaces need to be well understood. This is important to correctly achieve the desired functionality. Such dependency can be potentially handled by various match-making mechanisms. However, in addition, depender web service may need to guarantee various non-functional requirements (NFR) for it to be a desirable candidate to be involved in the composition. Some examples of NFR include availability, reliability, response-time, and accessibility. These QoS dependencies become even more crucial when the tasks exhibit fit and sharing dependencies. Furthermore, when third party web services are accessed remotely, security

becomes an important consideration. Finally, protocol mediation dependency arises if the end-points of the involved web services are not all based on a specific protocol such as SOAP.

Conceptual Model

Representing task dependencies in web services is the first step in capturing rationale behind web service composition. A variety of schemes for representing dependencies have been proposed in the literature. Following prior research in the area of design rationale management (Ramesh and Dhar, 1992), we use rules to represent task dependencies. In fact, business rules are commonly used to represent dependencies among tasks. A business rule may be defined as a guidance or an obligation concerning conduct, action, practice, or procedure within a particular activity or domain. It is a statement that defines or constrains desired logic of the business. Business rules provide the definition of a business and business concepts. They also allow us to know what should be ensured, permitted and prohibited (BusinessRulesGroup, 2004).

Business rules control or influence various aspects of the business (Gould, 2000) such as when to restock inventory, whether to extend credit to a customer and by how much, and whether to apply sales tax in a sale transaction. Business rules-driven approaches are getting increasing attention due to (1) maturity of emerging technologies such as web services, (2) dramatically falling costs of the IT processing power needed for rules automation, and (3) the need for a flexible and prompt response to changing market requirements. Various web services usage scenarios (Anderson, Chapman, Goodner, Mackinaw and Rekasius, 2003) suggest that the composition of web services is affected by user activities and other web services that are in turn governed by business rules. For example, the selection of a product or a delivery option affects the composition of the subsequent dependent web services.

We have developed a conceptual model, shown in Figure 1 that illustrates the dependencies among the tasks for the web services composition regarded from a business point of view. The shaded area represents the scope of the conceptual design activities addressed in this paper. Business rules provide the overall context in which tasks are to be performed and govern the task execution. We use this conceptual model to guide the design for our prototype Web Services Manager (WSM) that supports the management of business rules and task dependencies in the web services composition. This prototype can be used to instantiate various building blocks of the conceptual model. An e-commerce scenario for online shopping presented later demonstrates how business rules and task dependencies are used in the process of the web services composition.

In the next section we demonstrate how dependencies are expressed in BT-DML, a XML-based markup language. Motivation behind this XML-based representation comes from the fact the web-services use XML messaging specifications and to effortlessly integrate of such dependency knowledge with other composition protocols.

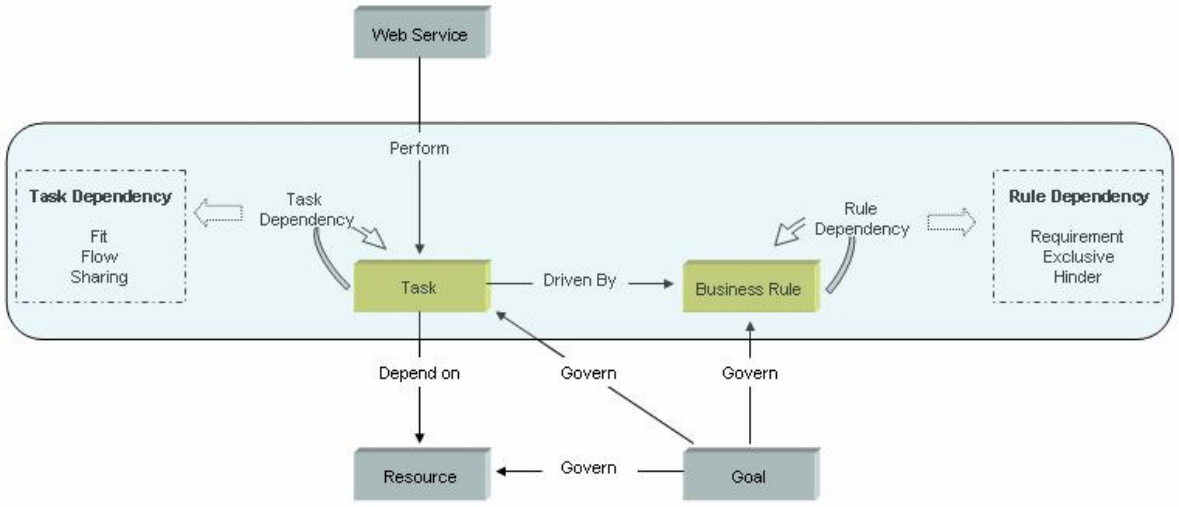

Figure 1: Conceptual model for web services composition

SPECIFICATION OF DEPENDENCIES

To enable better service coordination, discovery and classification (Tolksdorf, 2002), we express dependencies in XML format so that they can be utilized in the web services composition. The need to express rationale behind critical design

decisions is well recognized (Spanoudakis, Zisman, Perez-Minana and Krause, 2004). With an XML based representation, dependencies among tasks can be created and shared even by tools that do not interoperate. Also, dependency information can be recorded hierarchically. XML with a well structured schema has power to provide customized views of dependency information to suit the interests and requirements of different stakeholders.

We define BT-DML, a Business Rule and Task Dependency Markup Language that consists of business rules description, task descriptions, and a set of dependencies among tasks. The basic structure of BT-DML is presented in Figure 2 in terms of business rules and tasks.

– Business rule: we represent its type, name, target, and its dependencies with other business rules. When a business rule has a relationship with another business rule, it can specify its target business rule and relationship in terms of dependency. Dependency classification proposed by Bühne, Halmans and Pohl (2003) is employed in the representation of business rules dependency (Kim and Jain 2005). When a business rule has several dependency relationships with multiple business rules, these multiple dependencies can be recorded using name and BR-Dependency tags sequentially.

– Task: the central structure is similar to business rule element except for a few elements (strength and subtask) and an attribute (description). Strength element is used for showing the strength of dependency. For example, if a business rule is enforced by a regulation, the strength is *strong*. If related tasks need to be changed, the system using BT-DML will alert users. A subtask element will be used for representing a hierarchical structure among tasks. In case of task dependency, the dependency among tasks can be chosen among three task dependencies. By using an attribute called *TargetBR*, a task element can be linked to a business rule element. In addition, the coordination mechanism can be shown in a description attribute. Although the basic flow can be expressed by capturing dependency among tasks, the actual coordination mechanism based on the dependency among tasks is selected among various coordination mechanisms. A system that uses BT-DML can generate several coordination mechanisms based on the specified dependency.

.

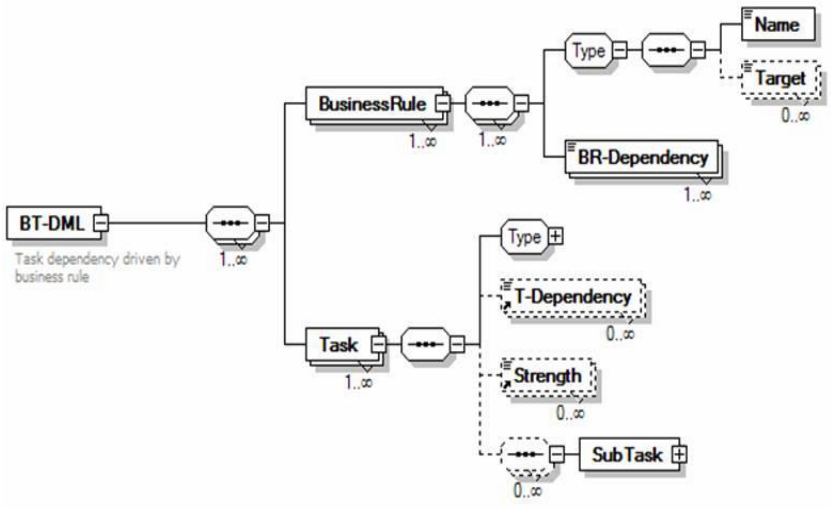

Figure 2: Structure of BT-DML

BT-DML enables a rich representation of task dependencies. It provides a way to describe task dependencies at an abstract level with a hierarchical structure. With the use of XSL, dependency information expressed in BT-DML can be reviewed by developers involved in web services composition. With this information, discovery and coordination of web services can be improved. For space constraints, the complete BT-DML schema is not included in this paper and it is available from the authors upon request.

A SCENARIO INVOLVING COMPOSITION ACTIVITIES

In an online shopping scenario, consumers normally browse through the online catalog to view various products/services offered by an online business before they make any purchases. Basic tasks involved in such a scenario typically include determining consumer type (revisiting user, first-time user, government, etc.), providing check-out, and shipping and delivery services. Determination of the delivery date depends upon the shipping method, availability of the product, and the weather at

both the start and destination location (assuming destination zip code is known beforehand). There are two possible scenarios in which a consumer makes an online purchase. The first scenario assumes the basic shopping process where before a consumer can purchase an item, he/she needs to log-in. Also in this scenario, consumer may not know the approximate delivery date and total price unless he/she logs-in and complete various activities within the check-out task. Table 1 shows this basic scenario and detailed breakdown of these various activities. A web service that supports this scenario should be composed to provide these various functions in the order specified in the scenario.

Sequential task on the basic scenario	Sub-tasks within the check-out task	Sub-tasks within the delivery data determination task
1. Browse through the online catalog		
2. Add items to the shopping cart		
3. Complete log-in procedure		
4. Complete check-out procedure	4.1 Provide shipping method & rate	
	4.2 Determine the delivery date	4.2.1 Shipping method
		4.2.2 Product availability
		4.2.3 Weather check
	4.3 Determine the total price	
	4.4 Verify the payment details	
5. Ship the items		
6. Bill the customer		

Table 1: Basic scenario

Though the above scenario is common in e-commerce websites, specific business situations may require variations in the way these functionalities are supported. For example, some recent studies that examine online purchasing behavior of the users suggest that making the process more 'effort-saving' can result in the increased ratio of buyer to visitors (Cho, 2004). Examples of such changes include provision of approximate delivery date and total price even before the consumer logs-in. Having such information available without going through check-out is likely to result in higher probability of a sale. This suggests changes in the basic sequence of tasks. In the new scenario, the delivery date and total price are provided when the consumer is browsing through the online catalog. This requires changes to the way the web services that support the check-out task are composed. For example, calculation of delivery date and price will need to be done at the time of browsing. Also during the check-out, these selected options may be redisplayed if the consumer needs to make any change and enter coupon codes (if any).

In Figure 3, we show how this scenario can be represented in BT-DML. Two business rules – 'Display Delivery Date During Checkout' and 'Enable Shipping Method Selection' - are associated with each other. Specifically the former rule is dependent on the latter through requirement dependency. DeliveryDateCalculation Task exists under a DisplayDeliveryDateDuringCheckout business rule. This task has three sub tasks: ShippingMethodSelection, ProductAvailabilityCheck and WeatherCheck. Both subtasks have fit dependency with its upper-level task: DeliveryDateCalculation. The strength of dependency between first two sub tasks and an upper-level task is critical because these two sub tasks are essential to calculate a delivery date. However developers may assume that WeatherCheck sub task is optional to perform delivery date calculation task in this case.

```
<BusinessRule>
    <Name>DisplayDeliveryDateDuringCheckout</Name>
    <Target>EnableShippingMethodSelection</Target>
    <BR-Dependency>Requirement</BR-Dependency>
</BusinessRule>
<Task TargetBR="DisplayDeliveryDateDuringCheckout">
    <Name>DeliveryDateCalculation</Name>
    <SubTask>
        <Name>ShippingMethodSelection</Name>
        <Target>DeliveryDateCalculation</Target>
        <T-Dependency>Fit</T-Dependency>
        <Strength>Critical</Strength>
    </SubTask>
    <SubTask>
        <Name>ProductAvailabilityCheck</Name>
        <Target>DeliveryDateCalculation</Target>
        <T-Dependency>Fit</T-Dependency>
        <Strength>Critical</Strength>
    </SubTask>
    <SubTask>
        <Name>WeatherCheck</Name>
        <Target>DeliveryDateCalculation</Target>
        <T-Dependency>Fit</T-Dependency>
        <Strength>Open</Strength>
    </SubTask>
</Task>
```

Figure 3: Usage of BT-DML

WSM Prototype functionality

In order to support the above scenario, we have developed WSM, a prototype system that provides the following functionalities:

1. A graphical interface for specifying the dependencies among the various artifacts, tasks and business rules helps capture the traceability across various knowledge components used in web services composition.
2. A dependency management mechanism helps identify the ramifications of changes that are to be supported in a changed scenario. For example, when the order of tasks specified in a business rule need to be changed, the components of web services that are affected and need to be changed are readily identified.
3. Mechanisms for supporting the semantics of certain commonly occurring dependency types are provided. For example, when the dependency type 'excludes' connects two artifacts (tasks, business rules etc), the system will ensure that the connected tasks may not be concurrently included during the composition of a web service.

Figure 4 shows the screenshot of WSM prototype which instantiates various primitives defined in our conceptual model defined in Figure 1. Figure 4 also illustrates the results of deliberations in the web services re-composition scenario. Using the graphical interface, information such as the stakeholder managing business rules, and dependencies among the business rules and tasks is captured. The stakeholder 'Smith' manages various business rules. Current business rule 'DisplayDeliverDateDuringCheckout' is dependent upon 'ShippingMethodSelection'. The tool allows representing also how the web services performing various tasks are dependent upon each other. For example, there exists an interface dependency between web services 'WSDeliveryDateCalculation' and its constituent web services 'WSShippingMethodSelect', 'WSWeatherCheck', and 'WSProductAvailCheck'. During the re-composition, dependencies among the tasks are addressed using dependency management module to identify the ramifications of changes supported in the modified scenario. For example in this modified scenario, fit dependency between the web services performing the task of 'determine the delivery date' and the web services performing various sub-tasks such as checking product availability, shipping method, and weather check is preserved. Figure 4 shows this change in business rule and how it affects the various tasks that are dependent on it. Further, it shows how the dependencies between the tasks 'DeliveryDateCalculation' and its subtasks 'ShippingMethodSelection', 'CheckProductAvailability, and 'WeatherCheck' are preserved even though these tasks are now performed before the user logs-in. This scenario illustrates how the rationale behind the selection and composition of web services can be acquired and used in WSM. The system can help developers cope with frequent, complex composition of web services.

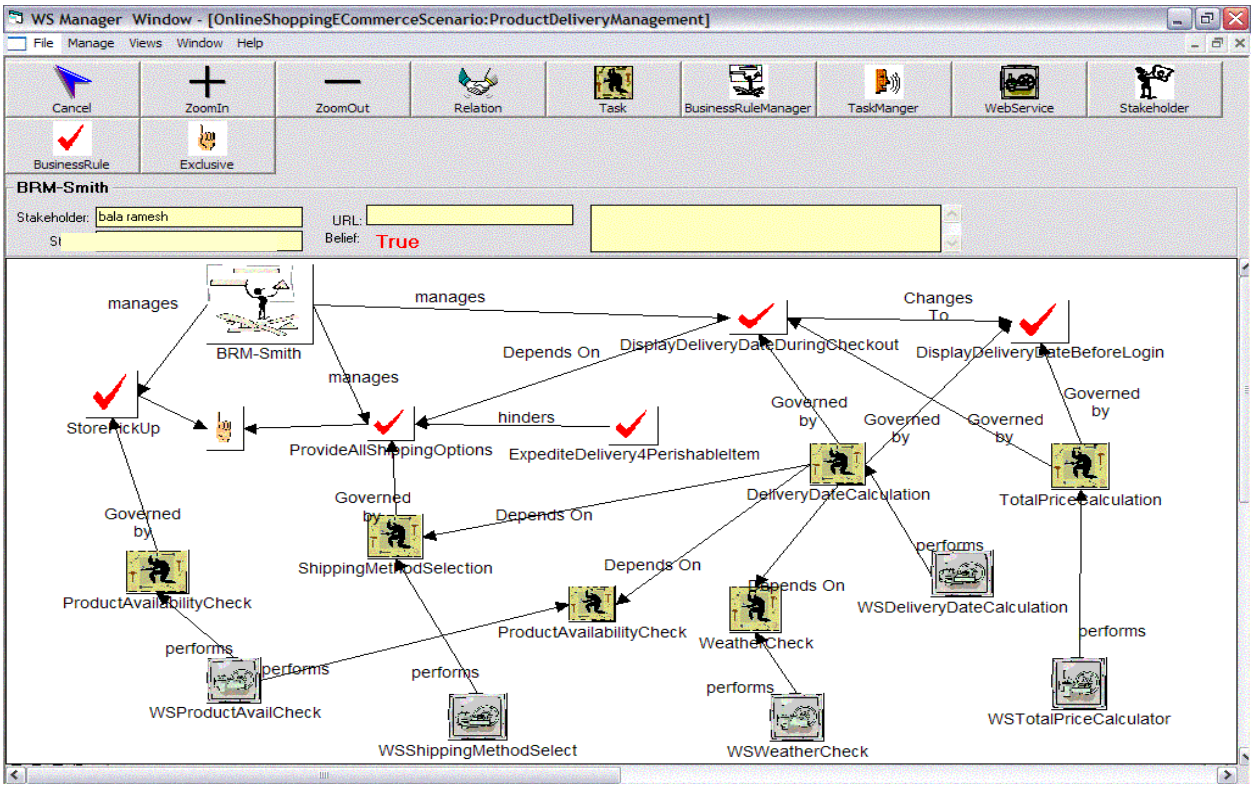

Figure 4: Managing business rules and task dependencies using WSM prototype

In summary, the prototype supports web services composition by providing the functionalities identified earlier:

- It helps developers understand the rationale behind a composition from a business perspective such that they can make informed design decisions quickly to respond to changing business requirements.
- It helps designers track the composition and hierarchy of tasks and related rules
- It helps designers identify the repercussions of changes to compositions necessitated by changes in requirements.

CONCLUSION

We support web services composition by defining a management layer, which is located above the implementation layer, by representing and reasoning with knowledge about dependencies among tasks and business rules.

The primary contribution of our work is the support for acquiring and using task dependency information to reduce the complexity in the dynamic web services composition. Specifically, our contributions include:

- The definition of BT-DML language that can be used to express various dependencies in XML-based format for achieving better service coordination.
- The development of a prototype (WSM) that helps developers to cope with frequent and complex composition of web services.
- The demonstration of the feasibility of our approach using scenario in the domain of online shopping.

Future research will include a rigorous evaluation of our approach and prototype integrated with BT-DML. We will evaluate our approach by several criteria such as effort and quality of composition. Composition effort will be measured by time taken to make composition and perceptive measures. Composition quality will be measured by evaluating performance in terms of completeness and accuracy. The evaluation will examine how rich description of dependency relationship can help human cognitive processes in web services composition and its management. A survey and lab experimentation for such an evaluation using both qualitative and quantitative data is currently underway.

ACKNOWLEDGEMENTS

This research was supported in part by the Ministerio de Educacion y Ciencia under project TIN 2005-06053 and a research grant from the Robinson College of Business, Georgia State University.

REFERENCES

1. Anderson, S., Chapman, M., Goodner, M., Mackinaw, P., and Rekasius, R. "Supply Chain Management Use Case Model, http://www.ws-i.org/SampleApplications/SupplyChainManagement/2003-12/SCMUseCases1.0.pdf," Web Services-Interoperability Organization.
2. Aral, S. "Automating Orchestration: Bridges toward Semantic Web Services (Draft version) Available at http://web.mit.edu/sinana/www/SemanticWebServices-SinanAral.pdf."
3. Baligand, F., and Montfort, V. (2004) A concrete solution for web services adaptability using policies and aspects, Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC'04), ACM Press, 134-142.
4. Bedford, T., and Cooke, R.M. (1997) Reliability methods as management tools: dependence modelling and partial mission success, *Reliability Engineering & System Safety*, 58, 2, 173-180.
5. Bühne, S., Halmans, G., and Pohl, K. (2003) Modelling Dependencies between Variation Points in Use Case Diagrams, Ninth International Workshop on Requirements Engineering: Foundation for Software Quality. In conjunction with CAiSE'03, Klagenfurt/Velden, Austria.
6. BusinessRulesGroup "Defining Business Rules...What is a Business Rule? Available at http://www.businessrulesgroup.org/brgdefn.htm."
7. Cho, J. (2004) Likelihood to abort an online transaction: influences from cognitive evaluations, attitudes, and behavioral variables, *Information & Management*, 41, 7, September, 827-838.
8. Choi, I., Park, C., and Lee, C. (2002) Task net: Transactional workflow model based on colored Petri net, *European Journal of Operational Research*, 136, 2, 383-402.
9. Colman, A., and Han, J. (2005a) Coordination Systems in Role-Based Adaptive Software, 7th International Conference, COORDINATION, Springer Berlin / Heidelberg, Namur, Belgium.
10. Colman, A., and Han, J. (2005b) Using Associations Aspects to Implement Organisational Contracts, Proceedings of the 1st International Workshop on Coordination and Organisation (CoOrg 2005).
11. Curbera, F., Khalaf, R., Mukhi, N., Tai, S., and Weerawarana, S. (2003) The next step in web services, *Communications of the ACM*, 46, 10, 29-34.
12. Dellen, B., Maurer, F., and Pews, G. (1997) Knowledge-based techniques to increase the flexibility of workflow management, *Data & Knowledge Engineering*, 23, 3, 269-295.
13. Gould, D. "Virtual Organization: Business Rules."
14. Hashimi, S. "Service-Oriented Architecture Explained, Available at http://www.ondotnet.com/pub/a/dotnet/2003/08/18/soa_explained.html."
15. Kim, J.W., and Jain, R. (2005) Web Services Composition with Traceability Centered on Dependency, HICSS '05. Proceedings of the 38th Annual Hawaii International Conference, System Sciences.
16. Ludwig, H., Dan, A., and Kearney, R. (2004) Cremona: an architecture and library for creation and monitoring of WS-agreements, Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC'04), 65-74.
17. Malone, T., Crowston, K., Lee, J., Pentland, B., Dellarocas, C., Wyner, G., Quimby, J., Osborn, C., and Bernstein, A. (1999) Tools for inventing organizations: Toward a handbook of organizational processes, *Management Science*, 45, 3, 425-443.
18. Mandell, D., and Mcllraith, S. (2003) Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation, the Second International Semantic Web Conference (ISWC2003), Sanibel Island, Florida.
19. McIlraith, S., and Mandell, D. (2002) Comparison of DAML-S and BPEL4WS, Knowledge Systems Lab, Stanford University, 2002.
20. McIlraith, S., Son, T., and Zeng, H. (2001) Semantic Web services, *IEEE Intelligent Systems (see also IEEE Expert Systems)*, 16, 2, 46-53.

21. Medjahed, B., Bouguettaya, A., and Elmagarmid, A.K. (2003) Composing web services on the semantic web, *The VLDB Journal*, 12, 4, September, 333-351.

22. Milanovic, N., and Malek, M. (2004) Ensuring Predictability and Correctness of Web Services Composition (Draft version), Available at http://www.informatik.hu-berlin.de/~milanovi/predictability-correctness.pdf, 2004, 1-8.

23. Peterson, D.M. (2003) Power to the BPEL: A technology for Web services, *Business Communications Review*, 33, 6, 54.

24. Ramesh, B., and Dhar, V. (1992) Supporting Systems Development by Capturing Deliberations During Requirements Engineering, *IEEE Transactions on Software Engineering*, 18, 6, 498.

25. Ramesh, B., and Jarke, M. (2001) Toward reference models for requirements traceability, *IEEE Transactions on Software Engineering*, 27, 1, 58-93.

26. Sabou, M., Richards, D., and Spluter, S.v. (2003) An experience report on using DAML-S, The Twelfth International World Wide Web Conference workshop on E-Services and the Semantic Web (ESSW '03), Budapest, Hungary.

27. Schulz, K.A., and Orlowska, M.E. (2004) Facilitating cross-organisational workflows with a workflow view approach, *Data & Knowledge Engineering*, 51, 1, 109-147.

28. Senkul, P., and Toroslu, I.H. (2004) An architecture for workflow scheduling under resource allocation constraints, *Information Systems*, In Press, Corrected Proof.

29. Spanoudakis, G., Zisman, A., Perez-Minana, E., and Krause, P. (2004) Rule-based generation of requirements traceability relations, *Journal of Systems and Software*, 72, 2, 2004/7, 105-127.

30. Stal, M. (2002) Web services: Beyond component-based computing, *Communications of the ACM*, 45, 10, 71-76.

31. Tolksdorf, R. (2002) A dependency Markup Language for Web Services, Net.ObjectDays 2002 Conference, Erfurt, Germany.

32. Yu, E., and Mylopoulos, J. (1994) Understanding Why in Software Process Modelling, Analysis, and Design, 16th International Conference on Software Engineering, Sorrento, Italy, 159-168.

33. Zeng, L., Benatallah, B., and Dumas, M. (2003) Quality Driven Web Services Composition, the 12th International Conference on the World Wide Web (WWW), ACM Press, Budapest, Hungary.