12-15-2012

Creating a Repository for the Design and Delivery of Web Services

John D. Delano
Cedarville University, jdelano@cedarville.edu

Atish P. Sinha

Hemant Jain

Creating a Repository for the Design and Delivery of Web Services

John D. Delano
School of Business Administration
Cedarville University
Cedarville, OH

Atish P. Sinha and **Hemant Jain**
Lubar School of Business
University of Wisconsin-Milwaukee
Milwaukee, WI

Abstract

Existing web service repositories are not well suited to handle the multi-layered representation of web services, nor are they able to support multiple development methodologies. We describe the design and development of a repository called Web Service Crawler that supports both a traditional development methodology for the initial design of services, and an agile approach for the design of composite applications. Web Service Crawler is developed based on a set of theory-based design characteristics, and includes novel facets to represent multi-layered web services, such as workflow, composition, and layer. The positive evaluation results indicate that Web Service Crawler can be used to guide web service designers, as well as enable consumers to more easily find and use services.

Introduction

As the adoption of service-oriented architecture grows, reuse plays an increasingly important role in the development of modern applications. Repositories that store and provide access to software components are key enablers of reuse. Existing component repositories have demonstrated that they can manage the task of storing and retrieving components with high levels of accuracy, precision, and recall (Vitharana et al. 2003). With a shift toward service orientation, however, existing component repositories and existing web service technologies, such as UDDI, may not be able to satisfy the demands of the new paradigm. Two factors primarily contribute to this inadequacy. First, existing technologies are not well suited to handle the multi-layered representation of web services. Second, service orientation introduces the challenge to support both the selection and composition of services to form new systems and the initial design of services to encourage future reuse.

These two factors suggest the need for a repository that can support both a traditional development methodology with a focus on long-term planning for the initial design of services, and an agile approach for the design of composite applications (Haines and Rothenberger 2010). However, it is not clear from the extant literature how to design a repository that supports the two different methodologies. Our paper addresses this issue by adopting the design-science research paradigm (Hevner et al. 2004) to create and evaluate a prototype repository based on a set of theory-guided design characteristics.

Web Service Crawler Design Specification

Designing a repository to fulfill the dual role of supporting design and delivery of web services requires a solid foundation in conceptual modeling as well as in human information processing. Although there is some overlap, the conceptual modeling literature primarily impacts the repository with respect to the initial design of services, while the human information processing literature impacts the repository with respect to the creation of composite applications. We therefore began the development of our prototype repository, *Web Service Crawler*, with a thorough review of those bodies of knowledge. Based on the review, we identified ten design characteristics that we use to inform the development of Web Service Crawler. Because of space constraints, we briefly describe the design characteristics below.

Simulation Capable. Experienced designers often use mental simulations to confirm the validity of a design. The use of concept maps can be used to allow the designer to visually specify new service compositions or select existing compositions, while guiding the designer through a simulation of his or her design.

Usability. Repository users should be able to view their interaction with the repository as a seamless interface (Matook and Indulska 2009). The repository should enable more experienced designers to quickly interact with the repository, relying on visual cues to trigger activation of prior knowledge, but for service selection, the user interface should be simpler in nature by presenting intuitive visual cues to enable easier exploration of the repository.

Queryability. Users can query the repository based on values stored in facets. The repository can accommodate both keyword-based and facet-based search strategies. Locating web services in the repository should not rely solely on a simple string-matching algorithm, but it should introduce some method for matching semantically similar services.

Orientation. Because the repository has to support the initial design of web services as well as the selection of existing services for composition, it must be designed to support two distinct audiences: service designers and service consumers.

Feedback. The web service repository should provide feedback relating to the cost of web service selection and the complexity of web service composition.

Understandability. A faceted approach to service representation can influence the understandability of each service stored in the repository by prompting the designer to consider specific attributes of each service that focus not only on current requirements, but also on future reuse potential. A faceted approach should also improve understandability of web services during search and retrieval (Matook and Indulska 2009).

Flexibility. The repository should use an extensible, faceted classification system that can evolve with the web services technology (Matook and Indulska 2009).

Completeness. The repository should contain an appropriate number of facets within its schema, such that web services are represented as completely as possible, while minimizing the cognitive burden of the analyst (Matook and Indulska 2009).

Correctness. Uses some kind of mechanism to encourage the entry of correct facet values, using either formal grammar checking or peer review (Lindland et al. 1994).

Generality. Represents multiple layers of web services as steps along an abstraction continuum between business logic and application logic (Matook and Indulska 2009).

Design and Development of Web Service Crawler

We created the repository using a 3-tier distributed architecture. The client tier was developed using Microsoft Silverlight, the middle tier using Windows Communication Foundation (WCF), and the database tier using Microsoft SQL Server 2008. The conceptual development of the prototype is guided by the design characteristics discussed above. From Semantic Network Theory, we understand that human memory is organized similar to a graph data structure, where nodes that represent memory engrams are connected via arcs that represent a single activation path (Ashcraft 2002). We pattern the design of Web Service Crawler after this structure to enable the repository to fill the role of external memory. In this way, the repository is able to support and extend user simulations, fulfilling the first design characteristic. Specifically, Web Service Crawler views the basic unit of memory as a single web service, and represents each web service as a node within a graph data structure.

The repository implements two types of arcs to act as activation paths when retrieving services from the repository. One is the composition relationship. Whenever a service initiates or makes use of the services at the same or lower layer in the service hierarchy, the repository creates an arc between the parent and child service to represent the composition relationship between them. The second type of arc is for semantic relationships, which represent dynamic connections between services based on their similarity. The repository calculates semantic similarity based on the ranking score provided by SQL Server's Full Text Search (FTS).

As suggested by multiple design characteristics, a faceted classification approach is used to represent the meta-data about each web service in the repository. The facets identified by Vitharana et al. (2003) are modified and extended to capture the features of web services more precisely (see Table 1). In addition to modifying the workflow facet to include a visual diagram of the business process automated by the web service, we introduce two completely new facets in the repository – *layer* and *composition* – to support the representation of web service hierarchies.

Facet	Description
Name	Used as a means to identify a web service.
Layer	One of four possible layers: utility, entity, task, or orchestration (Erl 2005)
Description	Descriptive information to facilitate retrieval
Keywords	Elicit thoughts about the important characteristics of the service.
Operations	Defines the capabilities that are offered by each service.
Role	Defines what part the service might play in a composite application.
Rule	Defines high-level business rules that are enforced by the service.
User	Identifies the end-user of the service.
Workflow	Contains a workflow diagram for web services that are categorized at the task or orchestration level to provides an efficient means by which the analyst can achieve a more detailed understanding of the web service's functionality.
Composition	Represents each web service as an object node within a directed graph. Each edge that connects two nodes represents a relational link between the two objects: either child (i.e. composition relationship) or friend (i.e., conceptual similarity between the two services).

Table 1 – Facets Used in Web Service Crawler Repository

The user interface of the repository is created to guide the user through the process of designing web service meta-data, as well as to identify and select existing services for later composition. Web Service Crawler accomplishes this through the use of two separate modes: design and exploration. When in design mode, the repository enables a web service designer to manage a list of web services that have been created in the repository by the designer as well as other designers who belong to the same group. This group sharing mechanism allows a team of designers to collaborate on the creation of a web service composition hierarchy. A designer can create a new web service by navigating to a wizard-like series of screens that guide the user by prompting him/her for the facets of information described earlier. When the user hovers the mouse pointer over help icons that are next to each data entry field, a tooltip window appears containing example data for the given field. This helps to encourage completeness and correctness of the data stored within each facet.

On navigating to the next page, the user completes the workflow facet by creating a graphical process flowchart for the new web service (see Figure1a). The interface provides a navigation option to proceed to the final page, where the value for the composition facet can be entered. The composition facet records the relationship between two services when one of the services invokes the functionality of the other service (see Figure 1b). The service that initiates the invocation is referred to as the parent, while the recipient of the invocation is the child. For each service composition facet, only immediate child services are represented in the context of their relationship with the service being edited.

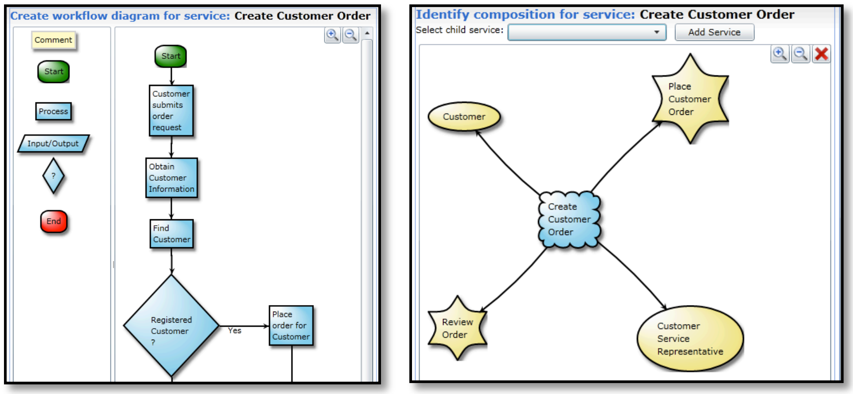

Figure 1a (left) – Web Service Crawler Workflow Screen; 2b (right) – Composition Screen

Web Service Crawler also provides an exploration mode for service consumers interested in building a composite application. It provides the ability to search for services in the repository using two search techniques: simple text search and facet-based search. Both techniques rely on SQL Server FTS to produce a relevance score for each search result, providing the user with valuable feedback. Each service listed in the search results provides a hyperlink to an exploration screen, where the user can visually examine the composition hierarchy for the selected service, along with the semantically related services in the repository (see Figure 2).

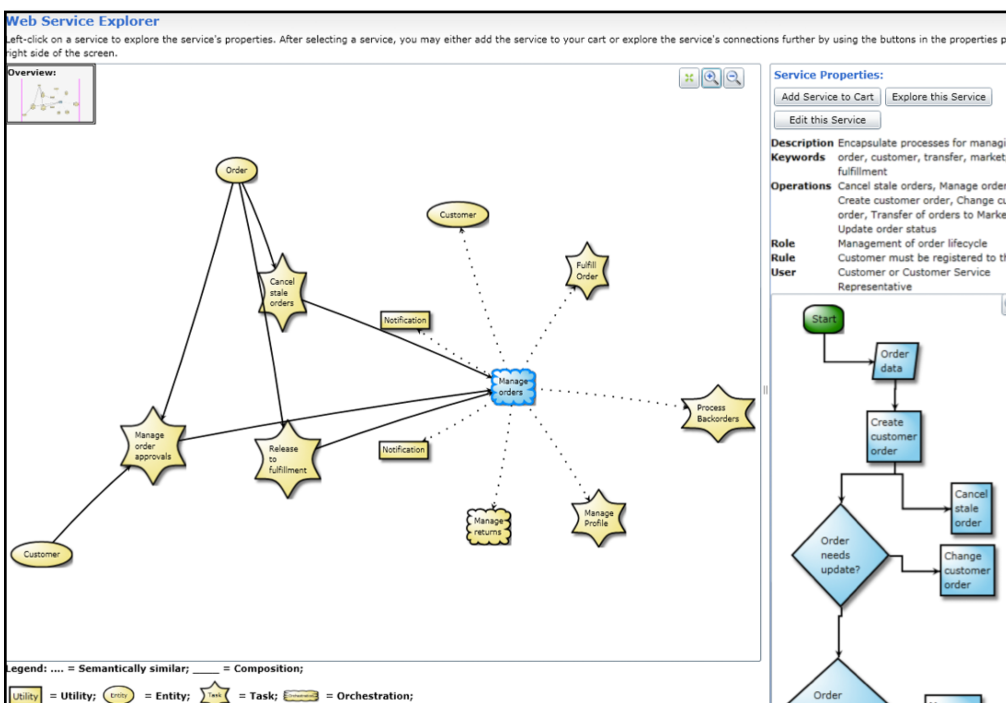

Figure 2 – Web Service Crawler Exploration Screen

The user can view the meta-data of a selected service on the right side of the screen. The exploration screen also provides a means for the user to “crawl” through the service linkages to view similar services or other services in the same composition hierarchy. The user can also add the selected service to a shopping cart. Once a service is inside the shopping cart, an indicator at the border of the screen displays information regarding the number of services in the cart, along with the total cost of the services, meeting the requirement of the feedback design characteristic.

Web Service Crawler Evaluation

We evaluated Web Service Crawler with respect to its role as a web service design tool and its ability to deliver web services. The validation process involved both the service designers who designed and stored the services in the repository and an independent panel of IT reviewers, serving as a proxy for service consumers, who were given a live demonstration of the system.

Two separate classes of students, enrolled in a Systems Analysis and Design course in two successive semesters, worked in teams to produce a design for a reusable web service hierarchy that consisted of at least 20 services. Students were asked to base their service designs on business rules that were publicly available at IBM's Infocenter website. The site contains high-level, hierarchical descriptions of e-commerce business processes.

The prototype was evaluated based on the "quality in use" HCI standard, which measures the degree of excellence to which a product meets the needs of its users. Its components include functionality, productivity, and usability. We developed an evaluation instrument from the standpoint of service design that consisted of 25 statements on a 5-point Likert scale. Sixty-five subjects participated in the evaluation (36 in the first semester and 29 in the second semester).

Exploratory factor analysis with Varimax rotation was utilized to demonstrate the discriminant validity of the factors represented by the questionnaire. The reliabilities for the usability, productivity, and functionality scales were 0.812, 0.778, and 0.779, respectively. We follow the convention of reporting usability on a 0 to 100 scale by normalizing the score for each component. The component scores for each of the two classes of students appear in Table 2.

	Service Designers			Service Consumers
Factor	Class 1	Class 2	Difference	
Usability	66.78	81.17	14.39 (t=4.435, p=0.000)	74.77
Productivity	71.18	72.43	1.25 (t=0.284, p=0.777)	73.24
Functionality	74.58	82.24	7.66 (t=1.999, p=0.050)	68.37

Table 2 – Quality in Use Scores by Factor

The scores for all the components are relatively high, especially given that human respondents tend not to rate any product at extreme ends of a scale. The scores provide strong evidence that the designers who used the repository found it to be of high quality in terms of functionality, productivity, and usability. These scores look particularly impressive when considering the complexity of the business processes for which the web services were designed.

An independent panel of reviewers validated the system from a service consumer perspective. Fourteen experts working for a major, global IT consulting firm and 50 IT students participated in the validation process. The Web Service Crawler system was demonstrated to both groups, and a questionnaire was then administered, similar to the one for designers, but from a consumer's perspective. Table 2 shows the average scores for the consumer group. The reviewer scores are quite high, providing support for the external validity of the repository.

When asked if a service workflow in the repository is helpful for understanding that service, the reviewers provided a high rating (74.206). When asked if composition of a service helps in understanding, the reviewers provided another relatively high rating (70.23). There is strong evidence, therefore, that the modified workflow facet and the newly introduced composition facet facilitate understandability of web services.

In addition to the objective scores, subjective feedback was also solicited from the reviewers on the strengths and weaknesses of the repository. With respect to the strengths, the recurring themes in the feedback were reusability, development speed, ease of use, and search capabilities. One reviewer stated, “Easy to use, nice GUI, good search capabilities, great idea – many companies suffer from reinventing the wheel every project despite efforts of developers to make code repositories that aren't supported by management.” As for the weaknesses of the system, most of the comments were related to implementation issues with the common theme being security; however, this was not the focus of the study.

Conclusion

This study contributes to the extant research in the area of storing, classifying, and locating web services. The number of web services that will be available to developers in the future will be extraordinarily high, and the need for an efficient classification, storage, and retrieval infrastructure will be essential for developers to assemble high-quality applications. The findings from this study indicate that the solution to that need lies in a lightweight, facet-based, web service repository that provides facets such as workflow, composition, and layer to support the design and delivery of web service hierarchies. From the standpoint of practice, the development of a repository that can be used to guide web service designers, as well as enable consumers to more easily find web services, should enhance the distribution of services within and across organizational boundaries. Unlike existing tools, such as IBM’s Service Registry and Repository, Web Service Crawler is lightweight so that it can better support the agile exploration and composition of web services.

Finally, it is important to address two limitations of this study. First, Web Service Crawler does not yet support the design and retrieval of web services based on non-functional characteristics, such as quality of service metrics. Future iterations of the repository will include this capability (Thurow and Delano 2010). Second, technical challenges prevented a hands-on evaluation of the repository by the service consumers. Future evaluations of the repository will include a more in-depth, hands-on evaluation of the system.

References

Ashcraft, M. H. 2002. *Cognition*, (3rd ed.) Prentice Hall: Upper Saddle River, NJ.

Erl, T. 2005. *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall: Upper Saddle River, NJ.

Haines, M. N., and Rothenberger, M. A. 2010. "How a Service-Oriented Architecture May Change the Software Development Process," *Commun ACM* (53:8), pp 135-140.

Hevner, A. R., March, S. T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research," *MIS Quart* (28:1), pp 75-105.

Lindland, O. I., Sindre, G., and Solvberg, A. 1994. "Understanding Quality in Conceptual Modeling," *IEEE Software* (11:2), pp 42-49.

Matook, S., and Indulska, M. 2009. "Improving the Quality of Process Reference Models: A Quality Function Deployment-Based Approach," *Decis Support Syst* (47), pp 60-71.

Thurow, N. A., and Delano, J. D. 2010. "Selection of Web Services Based on Opinion Mining of Free-Text User Reviews," in *ICIS 2010 Proceedings*: St. Louis, MO.

Vitharana, P., Zahedi, F. M., and Jain, H. 2003. "Knowledge-Based Repository Scheme for Storing and Retrieving Business Components: A Theoretical Design and an Empirical Analysis," *IEEE T Software Eng* (29:7), pp 649-667.