# What does Software Engineering Practice offer to Semantic Web Service Composition?

Bruce Spencer and Sandy Liu

National Research Council of Canada
{*Bruce.Spencer, Sandy.Liu*}*@nrc.ca*
*http://iit.nrc.gc.ca/il.html*

Bruce Spencer and Sandy Liu are the leader and a researcher in the Internet Logic group, which specializes in reasoning systems for Internet applications. The group's activities on the Semantic Web include (i) a Semantic Web Lab with close ties to RuleML.org, (ii) a graduate course at the University of New Brunswick on the Semantic Web Techniques, (iii) BASeWEB (Business Agents and the Semantic Web) workshop held annually in conjunction with the Canadian AI Conference since 2002, (iv) an open source reasoning engine jDREW [2] on SourceForge, (v) a queuing inference engine [3], (vi) DeFleX, an XML Router for agile knowledge workflows [1], and (vii) WSIRD, a rule-based data integration engine between Web Services [4].

**Challenges:** Web Services can be seen as functional components that can be selected for composition to achieve certain purposes, based on descriptions of the components' purposes, their preconditions and effects, and the data they accept and produce. The selection may be done by an abstract reasoner, such as a planner, which uses these descriptions to infer that a specific combination of Web Services will achieve a desired purpose. These services may not previously have been intended to work together.

Some proponents of Semantic Web Services optimistically envision that applications can be composed from distinct Web Services to operate on valuable data, execute reliably and provide accountability without much difficulty. Yet there is still a gulf between this vision and reality. Software engineering experience suggests that the development of reliable software requires following a methodology including analysis, design, testing, and validation. Since Web Service combinations will need to be reliable, lessons from software practice and experience should be intently applied to Semantic Web Services. How can this be done?

Software generated by humans is more likely to be reliable if it is developed from sound and well-documented designs, written according to coding standards, and subject to comprehensive testing. Since the Semantic Web Services compositions are expected to be composed by a machine, the design may not be documented, and the codes may not have been written according to coding standards. Certainly the combination will not have passed integration tests, although the individual Web Services are expected to have passed unit tests.

Much reliance is placed on the Web Service descriptions, which may be misplaced because there is no guarantee of accuracy and consistency. If the descrip-

tions of two Web Services do not directly relate to each other, a connection through a common ontology may be attempted, giving rise to another source of risk: the data models may be only partially integrated via the ontology. The underlying framework that calls the Web Services, such as a BPEL engine, must incorporate an error-handling strategy, which may rely somewhat on the ontology so that error messages and recovery actions can be meaningfully transmitted. Thus there are several sources of risk of software failure.

**Prospects:** Given a pair of Web Services where data from the upstream service flows into the downstream one, we suggest a monitoring component should be in place; it performs several functions: logs all data flowing through, responds to any error conditions from the downstream service, determines whether the upstream service should receive that error notification. It should also be configurable so that it responds to notifications of exceptions from other parts of the system to invoke error recovery procedures, such as rolling back financial transactions.

The logs could play a role in a post-runtime analysis of the Web Service, to ensure that their activities are appropriate. In the absence of integration testing, these logs become surrogate test cases and can be applied to do regression testing when a change occurs to a Web Service.

If semantic markup is available for both the upstream and the downstream service, the intervening component could use these descriptions to translate the data from the data model of the upstream service to the downstream one. Any descriptions of preconditions and effects could be given to the intervening component as conditions on the runtime data to be checked as it passes through. Error notifications expressed in the data model of one Web Service could be translated to become meaningful to another Web Service.

A precursor to such a intervening component, called an Inference Queue (IQ), is part of the WSIRD prototype [4]. The IQ conducts declarative XML information (facts) from its input to its output; it stores and runs rules to derive new facts, which are also emitted from the output. Its inference engine performs the logging, data translation and checking of preconditions and effects.

## References


1. C. Adsett, A. Bernardi, S. Liu, and B. Spencer. Realizing Weak Workflow with Declarative Flexible XML Routing in SOAP. In *Proceeedings of BASeWEB'04*. National Research Councial Canada, 2004.
2. B. Spencer. A Java Deductive Reasoning Engine for the Web. www.jdrew.org, 2004. Accessed 2004 Jan 12.
3. B. Spencer and S. Liu. Inference Quenes for Communicating and Monitoring Declarative Information between Web Services. In *Proceedings of RuleML-2003*, number 2876 in Lecture Notes in Artificial Intelligence, 2003.
4. B. Spencer and S. Liu. Inferring Data Transformation Rules to Integrate Semantic Web Services. In *The SemanticWeb - ISWC 2004*, number 3298 in Lecture Notes in Computer Science, pages 456–470. Springer, 2004.