Building Complex Systems of Web Services

László Gönczy

As use of Internet spreads dynamically in distributed applications there is a growing demand on loosely coupled systems. These integrate services running in heterogeneous environments of different providers without a knowledge of their implementation details. Web services architecture is the most current solution to this problem ([1],[2]). A Web service must use standard XML descriptions.

The main aspects of distributed environments - which should be covered by industry-standard languages or protocols - are the following (the most widespread solutions are given in brackets):

- Describing syntax of services - HOW to invoke it (WSDL).
- Discovering service interfaces - WHERE to find information (UDDI Registry, WSIL).
- Adding semantic description to the service - WHAT it exactly does. Several attempts were made to solve this challenge - such as DAML-S in the world of Semantic Web - but none of them has become mature yet [3].
- Messaging protocol - how to EXCHANGE messages (SOAP and industrial protocols such as JMS).
- Transport protocol - how to TRANSFER information. It is below the messaging protocol and can be almost any Internet protocol, not just one of the TCP/IP transport layer protocols (HTTP, FTP, etc.).

My objective is to examine how process composition, transaction handling, optimization, fault tolerance, reliability, error recovery, etc. of Web services can be ensured. There are some ad hoc initial XML-based specifications for these, but I would rather try to adapt proven methods from other fields of IT. My idea is to use principles, methods, and technologies which are proven in other environments such as:

- P-graphs for optimization. P-graphs (Process graphs, [4]) formulate a structural representation of problems where some inputs, outputs and operators are given. P-graphs are used at optimization of chemical industry processes (where inputs and outputs are materials, operators are distillers, mixers, etc.), and for processor diagnostic (where inputs are states of units, operators are decisions with their probabilities and output is test result [5]). According to my conception the inputs are input data, and the output is the result of service invocation where Web services (described by their interfaces) are the operators of the processes. Several algorithms are known to determine the combination network with minimal cost of used operators producing desired outputs of given inputs.
- System management via web services. There are a lot of system management tools available on the market. Their main problem is that the supported script-based failure recovery has to be implemented manually. If recovery actions were prepared automatically based upon a system model (given in UML or BPM) it would be possible to manage system components in a platform independent way using their Web service interface.

References

[1] K. Brown, R. Reinitz, "IBM WebSphere Developer Technical Journal: Web Services Architectures and Best Practices", `http://www-106.ibm.com/developerworks/websphere/techjournal/0310_brown/brown.html`

[2] D.A. Menascé, V.A.F. Almeida, Capacity Planning for Web Services, Prentice Hall, 2002.

[3] M. Sabou, D. Richards, S.v. Splunter, "An Experience Report on Using DAML-S" in Proc. of Twelfth International World Wide Web Conference Workshop on E-Services and the Semantic Web, 2003.

[4] Friedler, F., L. T. Fan, and B. Imreh, "Process Network Synthesis: Problem Definition", Networks, 28(2), 119- 124., 1998.

[5] B. Polgár, E. Selényi, "Probabilistic Diagnostic with P-Graphs", Acta Cybernetica, Vol. 16. p. 279-291 , 2003.