Requirements for Web Service Composition Management

Babak Esfandiari, Vladimir Tosic
{ babak, vladimir }@sce.carleton.ca
Carleton University, Ottawa.

I- Introduction

Although XML (Extensible Markup Language) Web services [Cer02] can be used for providing services to end users, the true power of Web Service technologies is leveraged through compositions (orchestrations and choreographies [Pel03]) of Web Services. Web services can be composed to achieve Enterprise Application Integration (EAI) within one company or to achieve business-to-business (B2B) integration of heterogeneous software and computing systems of different companies. The composed Web Services can be distributed over a network, running on different platforms, implemented in different programming languages, and provided by different vendors. The composition provides added value, either to end users or for further application-to-application (A2A) integration, when a Web service composition itself becomes a higher-level composite Web service.

In many cases, it is important to guarantee, monitor, and enforce quality, reliability and/or security of a composition of Web services, in ways similar to those used for individual Web services. But how to provide such guarantees when some or even all parts of the Web service composition are controlled by third parties, and distributed over the network? There is a need for Web Services Composition Management (WSCM). Given similarity in need for security, fault, accounting, performance and configuration management, we believe that Web Service Composition Management can roughly follow the same principles as network, systems and application management. Web service compositions are also implementations of business processes and workflows [Ley02], so their management can be related to business process management. However, because of the more volatile and dynamic nature of Web services and the lack of control over them, we believe that neither classical network and systems management nor business process management will be sufficient, and that a new paradigm is needed.

This paper is the result of our past experience in various domains which we believe will contribute to WSCM. Our work on the Web Service Offerings Language (WSOL) [Tos04b] and the Web Service Offerings Infrastructure (WSOI) [Tos03, Tos04a, Tos04b] explores specification, monitoring, and management of classes of service for Web services. It includes algorithms and protocols for dynamic adaptation of Web service compositions using manipulation of classes of service [Tos03].
But in addition to such direct work on Web service management, we believe some of our other experiences can also be applied to this domain. First, our group's ICARIS project

[Men01] researched run-time composition of electronic (Jini) services and encountered several topics that we studied in resent projects, notably the need for comprehensive service description and prevention of feature interactions. Therefore, we introduced techniques to detect feature interactions in Web services at the requirement analysis level [Wei04]. Further, our research group studied software "hot-swapping" (runtime software modification) in the context of telecommunication applications and network management [Fen01] with component technologies such as JavaBeans [Tan01], but also applicable to Web services. In addition, our work on schema-based peer-to-peer file sharing [Art03] and reputation propagation [Esf01] is derived from our interest in agent-oriented techniques for network management [Esf98].

While these projects addressed particular aspects related to the management of diverse electronic services and their compositions, they did not discuss an encompassing architecture for the management of Web service composition. In this paper, we analyze requirements for such a general Web service composition management system, discuss how our past results relate to these requirements, and outline approaches for future research and development activities.

II- Related Work

[Far02] is a survey of Web Service Management (WSM) issues and some possible approaches for their solution. While WSM is discussed from the application management perspective, the authors identify distinctive characteristics of Web Service technologies. They identify three WSM principles that they discuss in detail: 1) the use of separate management interfaces; 2) data collection by the run-time infrastructure, such as SOAP engines; and 3) the use of event collectors – entities (e.g., Web Services) to which various management events are sent. Further, several general Web Service management patterns are discussed. At the end of the paper, the management support of the IBM Web Services Toolkit (WSTK) is examined.

[Lay02] gives an overview on how Web Services relate to business processes and their re-engineering and management. When a business process is implemented with a Web Service composition, business process re-engineering is implemented with dynamic adaptation of this Web Service composition. Business process management (BPM) achieves monitoring of business processes, as well as enactment of business process re-engineering results. It can include management of contracts between participants. However, business process management tools traditionally do not automate management – they monitor business processes, present results on management dashboards, and then let humans perform management actions.

Our Web Service Offering Language (WSOL) [Tos04b] enables formal and unambiguous specification of classes of service, functional constraints, extra-functional (QoS) constraints, access rights, prices, penalties, and other management statements for Web services. It also provides mechanisms for the specification of static and dynamic relationships between different classes of service. The latter are used in our algorithms

and protocols for dynamic adaptation of Web service compositions using manipulation of service offerings [Tos03]. The corresponding Web Service Offering Infrastructure (WSOI) [Tos03, Tos04a] enables monitoring of Web services described in WSOL and implements our dynamic adaptation algorithms and protocols.

WS-Policy [Hon03] is a general framework for the specification of policies for Web Services. A policy can be any property of a Web Service or its parts. WS-Policy has a number of good features, such as flexibility and extensibility. However, the details of the specification of particular categories of policies will be defined in specialized languages. The only such specialized language currently developed is WS-SecurityPolicy. It is not clear whether and when some specialized languages for the specification of QoS policies, prices/penalties, and other management information will be developed. Further, WS-Policy does not detail where, when, and how are policies monitored and evaluated.

Perhaps the most relevant work is [Mac02], which discusses an overlay network for Web Service Management. The authors argue that distributed protocols must be introduced to deal with the federated management of 'relationships' between services. They mention discovery and selection of partners as some of the issues that can be dealt with through Service Level Agreement (SLA) management. Specific protocols are discussed, such as life-cycle, measurement and assurance protocols; the latter provides support for a rating service. The work presented in [Mac02] is related to HP's proposed language for the formal, detailed, and precise specification of SLAs for Web services [Sah02a] and to the corresponding infrastructure for SLA monitoring [Sah02b].

In the same vein, IBM has developed the Web Service Level Agreement (WSLA) framework [Kel03, Lud03] for the XML specification, monitoring, and management of custom-made SLAs. It defines the WSLA language and several tools for creation, deployment, and compliance monitoring of SLAs, e.g., WSLA Compliance Monitor.

III- The Need for Web Service Composition Management

We view *Web Service Management* (WSM) as management of particular Web Services or groups of Web services within a particular domain of management responsibility. For example, Web services provided with one application server might be managed as a group. Similarly, Web services provided by the same business entity might be managed in a unified manner as a group, either completely or only in some aspects. We see important similarities between WSM and traditional approaches to application, system, and network management. The traditional functional areas [Ste95] of performance, security, configuration, fault, and accounting management can also be identified for Web services. When Web services are pure software entities, they can be managed similarly to other applications, using adaptations of some existing application management solutions, such as Java Management Extensions (JMX) [Sun04] or Application Response Monitoring (ARM) [Ope04]. When Web services include some hardware or networking functionality, they can be managed using adaptations of some existing system management and/or distributed system management solutions, such as the Desktop

Management Interface (DMI) [DMT04a] and CIM/WBEM (Common Information Model / Web Based Enterprise Management) [DMT04b]. There are already a number of industrial products, such as HP's Management Integration Platform or Infravio Ensemble, and research projects, such as [Sah02b, Kel03, Shr03, Bro03], that address different WSM topics. These works are often focused on one functional area, such as performance or security management.

As an addition to WSM, we define *Web Service Composition Management* (WSCM) as the management of Web service compositions. Web services and Web service compositions can be viewed, and managed, from the technical and from the business aspect. From the technical aspect, Web service compositions are distributed computing systems. From the business aspect, they represent business processes and business products. Consequently, WSCM is positioned between traditional systems and network management on one side, and business process management on the other.

To achieve WSCM, a lot of work will have to be delegated to WSM managers of composed Web services and/or third parties specialized for particular WSCM activities. However, explicit definition and description of a Web service composition may not exist - a Web service composition may be the result of several independent bilateral or multilateral agreements between the composed Web services. Even when there is such explicit definition and description, centralized management of a Web service composition may, but need not exist. As a result, there might be a complex, multi-level, network of various management entities that must not only communicate, but also cooperate. This is a major difference between WSCM and traditional business process management. Existing workflow and business rule engines do not address this complexity of multi-party, multi-level, and multi-aspect management. Further, it might happen that not all participating businesses will be willing to relinquish control over their Web services to other participants or to outsource management activities to some trusted party. One of the key differences between WSCM and existing distributed systems and network management solutions is this presence of multiple domains of control with their own management goals. For example, a situation where every participating business wants to maximize its revenue and does not strive to accommodate analogous goals of its suppliers and consumers is not unrealistic. Further, management of the Internet infrastructure is a very challenging task. Consequently, WSCM will often not be able to involve full WSM management of the composed Web services and management of the Internet communication infrastructure. Therefore, the emphasis in WSCM should be on decisions related to which Web services are composed and how they interact, e.g., what contracts (e.g., Service Level Agreements – SLAs) are used between them.

Note that [Pel03] classifies Web service compositions into orchestrations and choreographies. In an orchestration, one party controls the order of the execution of the participating Web services. On the other hand, a choreography is a collaborative exchange of messages between the participating Web services. The differences between orchestrations and choreographies impact their management. For example:

- Orchestrations often have higher degree of control over the composed Web services and can invoke more WSM actions.

- Willingness to share management information is often higher in orchestrations than in choreographies.
- Centralization of management activities is often more appropriate for orchestrations than for choreographies.
- Root cause analysis is often easier for orchestrations, due to the existence of explicit descriptions of the composition.
- Trust seems to be a bigger issue for choreographies than for orchestrations.

However, these and other distinctions cannot be definitive. It is possible to find counter-examples for any of the above or similar statements. Therefore, we think that it is appropriate to design a general WSCM system that can be used for both orchestrations and choreographies.

IV- Requirements for Web Service Composition Management

We argue that WSCM extends WSM and integrates it as part of a larger lifecycle. When a WSM system detects that a monitored system is misbehaving or violating an SLA in a way that cannot be resolved at the WSM level, WSCM software takes over, as shown in Figure 1. WSCM software can first attempt to discover services with a similar functionality. The next step is then to narrow the selection down to a single service, using various criteria such as quality of service and reputation. It is also important to detect whether the new composition can potentially trigger undesirable feature interactions. Finally, the re-composition must be initiated and completed seamlessly, without shutting down the application. In some cases, it is possible to bypass some of these steps (as shown by the dashed lines in Figure 1). For example, when alternative Web services were already discovered at some earlier time, the service discovery step can be bypassed.

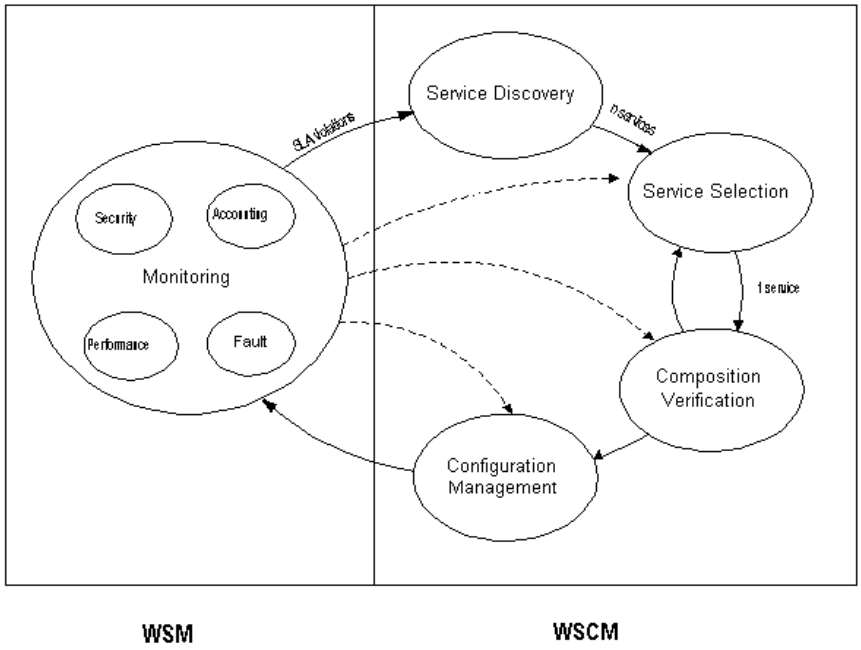

Figure 1: The WSCM Lifecycle

In the next subsections, we detail each step of the lifecycle, its requirements, as well as proposed methods for addressing these requirements.

IV.1- Service Discovery

The initial composition of Web services is one of the first aspects that have to be covered by WSCM.

Requirement 1.1: Support Web service discovery, in a centralized and/or peer-to-peer manner

First there is a need to discover available services. Web services are usually discovered by querying registries using interfaces such as UDDI.

While registries can be a convenient way to discover services, their centralized nature can lead to fault-tolerance, performance, and bias issues. One way to deal with such issues would be to envisage, similarly to network management, a broadcast discovery protocol. However, broadcasting at the application level could waste network bandwidth.

We are currently working on an alternative based on peer-to-peer service advertisement. In U-P2P, our peer-to-peer file-sharing framework [Art03], communities are created based on the specification of the schema of the files they share as well as the deployment model they use. This allows for:
- more effective search, as the schema provides structured metadata that can restrict the search domain to similar files.
- more efficient and scalable search, as the central registry bottleneck can be eliminated in favor of a Gnutella-type query propagation (fig 2).

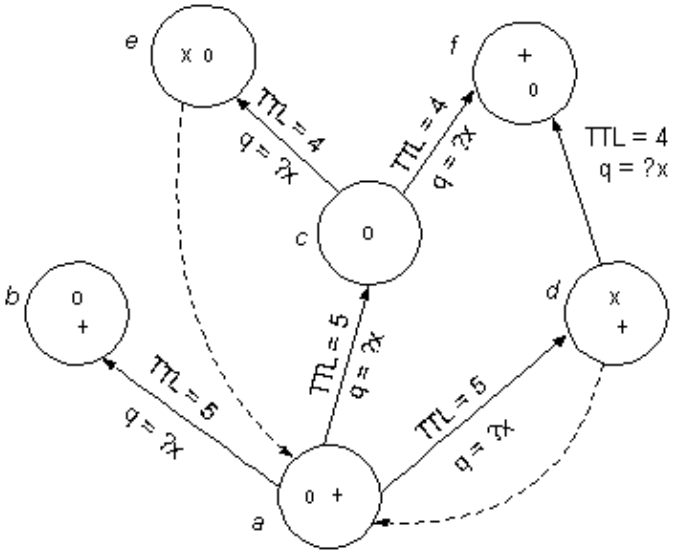

Figure 2: Gnutella routing used to discover service x with a time to live (TTL) of 5

Requirement 1.2: Support syntax, semantic, and pragmatic (business) matching of Web services

To apply the above mentioned concepts to Web service discovery, we first need a schema that describes the characteristics of the service adequately. The adequacy of a given Web Service depends on many aspects: in particular the *syntactic match*, the *semantic match*, and the *pragmatic (business) match*. The syntactic and semantic matches help discovery by using a black-box view of services, whereas the business match supports the selection process using a glass-box view [Bat03].

Syntactic match should provide means to examine the service's API (usually described in WSDL), verify the mapping between the offered API and the client's expected one, and generate an adapter or proxy in case the mapping isn't satisfactory. Such adapters or proxies can take on other management roles as explained in the remainder of this paper.

The semantic match should go beyond a simple keyword match. Functional properties of the service must be described in an unambiguous, ideally formal, manner. Semantic Web efforts, such as DAML-S [Pao03] go in that direction. Further, UDDIe [ShA03] extends UDDI with support for specification of a bag of user-defined properties and search based on these properties. These properties can describe, for example, price or hardware requirements (used for Grid services). Web services can be queried with simple expressions that can contain Boolean operators ('AND', 'OR') and value comparisons (greater, less, equal). Note also that while the main goal of our WSOL is to support management of Web services and run-time adaptation of Web service compositions, the comprehensive description of Web services in WSOL can be used for easier discovery, comparison, and selection of Web services.

For the sake of efficiency, effectiveness, and objectivity, it would be unreasonable to expect the registry, or the queried services in a P2P deployment, to apply the business match. It is rather up to the querying service to make the evaluation. As such, we will discuss those aspects in the next section dedicated to service selection.

Requirement 1.3: Implement algorithms and protocols for the routing of Web service advertisement, search queries, and search results

Another ingredient required for a peer-to-peer discovery of web services is to enable them with the capability to propagate search queries to neighboring nodes. The neighborhood could for example consist of all Web services with which the given service is or was composed. Should each web service provide additional ports for peer-to-peer routing, or should this be the responsibility of the container, or finally should there be one router per ownership domain? We will come back to deployment issues in a later section of this paper.

IV.2- Service Selection

Requirement 2.1: Support evaluation, comparison, and negotiation of extra-functional characteristics of Web Services, such as QoS, trust, and reputation

The discovery of adequate Web services is only the first step. We assume that all found services satisfy functional requirements or at least can satisfy them with the use of available adapters. Next, narrowing the list down to the best candidate can be achieved by evaluating and comparing the business match, i.e. the extra-functional properties such as quality of service (QoS), trust, and reputation.

Requirement 2.2: Enable definition, selection, and negotiation of comprehensive monitorable and enforceable contracts between Web services

First of all, there is a need to agree on technical QoS aspects (response time, throughput, availability) and business QoS aspects (pricing, billing, warranty policies, and other aspects of "quality of business") offered by the Web service. Negotiation of quality of service, terms of payment, and other non-functional requirements have also been studied. The results of this negotiation are explicit contracts, e.g. service level agreements (SLAs). Several languages have been developed for the formal and precise XML-based specification of custom-made SLAs for Web Services, such as the HP work in this area [Sah02a, Sah02b], WSLA [Lud03, Kel03], and SLAng [Lam03]. In addition, service offerings in our WSOL [Tos04b] can be viewed as anonymous SLAs. HP's language is powerful and compatible with the standard Web Services Description Language (WSDL). An SLA consists of a description of the SLA's validity period, involved parties, and a set of Service Level Objectives (SLOs). Every SLO contains a description of days and times when the SLO is valid, as well as a set of clauses. A clause describes one or more items (e.g., operations) for which measurements are performed, times or events (e.g., operation completion) that trigger evaluation, measurement samples that are used for evaluation, one Boolean evaluation function, and an action that is performed if the evaluation function returns 'False'. An evaluation function is applied over measurement samples to calculate a QoS metric and to evaluate an appropriate condition. It captures definition of this QoS metric. SLAs in IBM's WSLA have relatively similar structure, while SLAs in SLAng are less powerful.

Requirement 2.3: Delegate monitoring of contracts (particularly SLAs) to appropriate WSM entities

Agreements such as contracts and SLAs should be stored and monitored by a WSM platform, either a party of the Web service composition, or a trusted third party. Any failure to comply should be detected. If possible, it should be addressed at the WSM level. Otherwise, the failure could lead to contract re-negotiation or even breaking the Web Service composition. We will discuss this aspect further in this paper.

Requirement 2.4: Model trust and relate monitored information at the WSM level with trust

A contract can also be used to affect negatively the possible future selection of that same Web service, or any Web service from that particular vendor. More generally, monitoring could support the establishment of trust relationships between vendors. In its most simple form, trust can be modeled as a function that, given two agents (in this case two Web services or vendors), returns a number between 0 and 1 [Esf98] where 0 would mean utter distrust and 1 represents complete trust. The number can for example reflect to what extent (i.e. how often) the QoS levels have been met by the service/vendor in the past. More complex trust models can also capture: the amount of confidence in the trust value (based on past observation and interactions), trust refined to a particular competence of the vendor, or other relevant information. For a study of the topic, see [Mar94].

Requirement 2.5: Address trust propagation to enable comparisons of previously unknown vendors and/or Web services

But how to compute a default value of trust when there has been no previous transaction with a given vendor or Web service? We can take a Social Network Analysis approach [WaFa94] here, and apply it to the problem of trust [Esf98]. Once one-on-one trust is established by a Web service with its peers, it becomes interesting to study the possibility of propagating it along a chain of acquaintances, to allow for trust acquisition based on second-hand (or more generally nth hand) information.

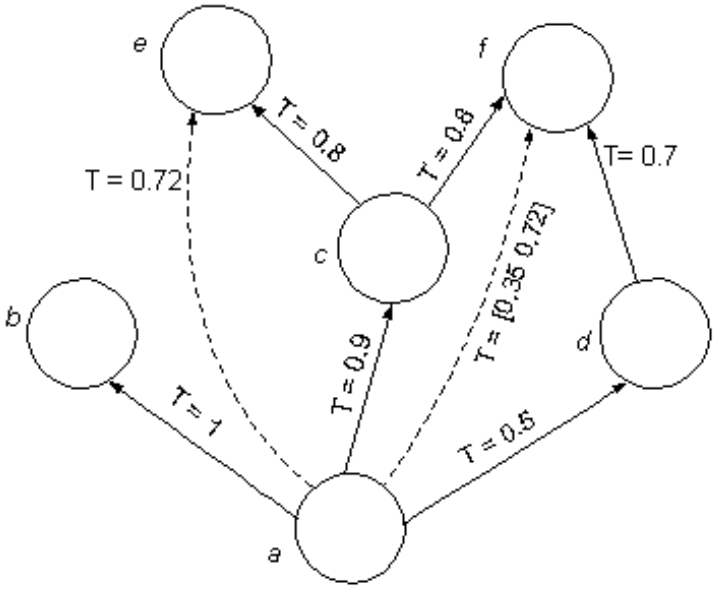

Figure 3: Using trust propagation to obtain a reputation value/interval for *e* and *f*

Let us consider a graph representing a network of Web services and their acquaintances, i.e. Web services or groupings of Web services per host. It is a directed, labeled graph, where an (a,b) edge represents the trust value that a has of b. Edges are absent where the trust value is unknown. Our goal therefore will be to determine those values (i.e. create edges), as well as update the value of existing edges, based on trust propagation. One way to do so is to multiply the trust value along the chain of trust:

```
Trust(a,c) = Trust(a,b) x Trust(b,c)
```

When different paths give different values, we calculate a trust interval, bounded by min-trust-product(over all paths without a cycle) and max-trust-product(over all paths without a cycle). In a distributed setting, where the graph's edge values are not centrally known, the problem of calculation of the trust interval becomes equivalent to the problem of routing in a communication network. Distributed algorithms such as RIP [Hed88] can be applied. The merit of such algorithms is also that they allow the regular and incremental update of trust values.

IV.3- Composition Verification

Requirement 3.1: Use comprehensive descriptions of Web services in service discovery and selection stages to discover and prevent, as much as possible, unwanted feature interactions

One often overlooked aspect of service composition is the *feature interaction problem*, which has been studied quite thoroughly in the telecommunications domain. Loosely speaking, feature interaction occurs when a given feature in a system does not behave the same as when in presence of one or more other features. A classical example in telephony is the interaction between Call Forward on Busy (CFB) and Call Waiting (CW). When both features are enabled by Alice, it isn't clear which feature should trigger when she is busy, unless for example a precedence scheme between features has been defined in advance [Uta01]. In the Web services world, personalization and privacy features can conflict if information is disclosed to a third party as a result of service composition.

Unexpected feature interactions cannot always be discovered and prevented before the composition, although using a comprehensive, if not formal, specification can help in their minimization. We have proposed a methodology in [Wei04] which can take advantage of a glass-box description of Web service processes using goal diagrams and use-case maps to detect and solve certain types of interactions. Roughly, the approach consists firstly in analyzing the diagrams to see whether the combination of features can impact negatively some extra-functional requirement as specified in the goal diagrams, to the point that a feature supporting it is broken. Refactoring the goal diagram (i.e. modifying it without changing its semantics) or the use case map can often help solve the problem. Typical refactorings consist of changing the ownership of services to avoid conflicts of interest, and breaking the "chain of responsibility" to deal with hidden assumptions. In the context of WSCM however, such refactorings can seldom be applied, as it would be hard to change the type of workflow on the fly. The detection methodology can still be used however, and compositions that trigger feature interactions would be rejected.

Requirement 3.2: Implement run-time detection, recovery, and resolution of feature interaction problems

There are also other mechanisms (e.g., using a pipe-and-filter architecture) to limit unexpected feature interactions. It should be studied to what extent are such existing mechanisms for minimization of feature interactions applicable for compositions of Web services. In addition, management solutions for detecting unexpected feature interactions after the composition was made and for recovering from such situations are needed.

IV.4- Configuration Management

As Web services cannot be deployed by their consumers (they are already deployed by their providers), deployment is not an issue or an option in configuration management of Web Service compositions. When a configuration of a Web service composition has to be changed to improve performance or to resolve a fault problem, Web services may have to be reselected and rebound, or contracts between them have to be re-negotiated.

Requirement 4.1: provide support for Web service "hot-swapping"

Given the high-availability requirements, such configuration changes cannot be made offline, hence the need to support mechanisms for dynamic upgrade, also known as hot-swapping. We have already explored issues specific to component hot-swapping [Tan01] and developed a hot-swap management platform for JavaBeans components. Such support allows the operator to:

- decide on the right timing for the hot-swap operation,
- create adequate mappings in case of interface mismatch between the old and the new service,
- solve state transfer problems between the same two services, and
- roll back to the old service in case a problem occurs.

If the upgrade is initiated by the service provider, it should be transparent to the consumer. As such, hot-swapping becomes a configuration management issue to be handled by WSM. On the other hand, if the upgrade is initiated by the service consumer, hot-swapping is a re-composition process involving at least two parties and should be handled by WSCM. The adapter/proxy components used in case of a syntactic mismatch can provide the extra level of indirection required to perform the hot-swapping task.

Requirement 4.1: provide support for Web service "hot-swapping"

A more lightweight approach that we are also investigating is the possibility to modify the QoS level offered by a service in order to meet new requirements [Tos03, Tos04a]. That would result in effect in a new configuration for the service. Obviously such an approach requires a formal description of the various QoS levels that are offered by the given service. Such a description should at least be compatible at the syntactic level with

WSDL, and at the semantic level build on shared ontologies defined for example in an RDF-based language such as DAML-S.

IV.5- Other Management Aspects

Alarm detection and correlation issues can be particularly tricky, as a WSCM platform might not have full control of the monitored Web services. In particular, some Web services might not reveal the services that they rely on in turn. So if one of those services fails, it would be difficult to determine the exact origin, unless other monitored parts are also affected, which might enable some degree of correlation. On the other hand, it seems that alarm detection and correlation can be addressed in such cases using propagation of fault detection and resolution responsibility through the Web services composition.

V- Deployment Considerations

What type of framework is suitable for WSCM? A centralized architecture operated by a third party could not only provide a registry for service discovery, but also a rating service based on feedback and direct observation in addition to certification capabilities. It could also act as a neutral intermediary for SLA negotiation and monitoring. Problems such as potential for bias, performance bottleneck and single point of failure can be obstacles to such a deployment.

A federated approach based on an overlay management network such as the one proposed by [Mac02] could address some of the above problems. There are ideal placeholders in such architectures for distributed service discovery and reputation propagation as described in this paper. This approach would however require bigger involvement in the management tasks from all Web service containers involved in service compositions.

The next level of granularity would correspond to relying on each Web service to provide its own additional port types and operations for management purposes. The dependency on the service container is thus eliminated, but the burden is now on the Web service developers (and development tools) to implement compatible management capabilities.

As can be seen there are trade-offs for each approach, as well as various opportunities for vendors and service providers.

VI- Conclusion

In this paper we proposed a definition for Web Service Composition Management (WSCM), in which we include Web Service Management (WSM) as an element of a bigger lifecycle. We discussed a specific set of requirements and provided an overview of possible approaches for implementing them, as summarized in the table below:

Requirement Area	Proposed Techniques
Service Discovery	peer-to-peer discovery
Configuration Management	software hot-swapping, classes of service
Composition Verification	feature interaction detection
Trust Management	reputation propagation

We believe in the promise of a federated approach to WSCM, especially since our proposed techniques would leverage such a deployment. Future work could therefore consist in collaborating with architects of such systems to evaluate the inclusion of our techniques in a realistic setting.

Another interesting area of work would be the study of the compatibility of our proposed techniques with grid services and relevant emerging standards, as discussed for example in [Fos02] and [Cza03].

References

[Art03] Arthorne, N., Esfandiari, B., Mukherjee, A., U-P2P: A Peer-to-Peer Framework for Universal Resource Sharing and Discovery, USENIX 2003 Annual Technical Conference, FREENIX Track, pp. 29-38

[Bat03] Battle, S., Boxes: black, white, grey and glass box views of web-services, online at: http://devresource.hp.com/drc/topics/web_services_mgmt.jsp (last accessed May 2004)

[Bro03] Brose, G.: Securing Web Services with SOAP Security Proxies. In Proc. of the 2003 International Conference on Web Services - ICWS'03 (Las Vegas, USA, June 2003). CSREA Press (2003) 231-234

[Cera02] Cerami, E.: Web Services Essentials. 1st edition. O'Reilly & Associates (2002)

[Cza03] Czajkowski, K., Dan, A., Rofrano, J., Tuecke, S., Xu, M.: Agreement-based Grid Service Management (OGSI-Agreement), Version 0. June 26, 2003. Global Grid Forum. On-line at: http://www.globus.org/research/papers/OGSI_Agreement_2003_06_12.pdf (2003)

[DAM03] The DAML Services Coalition: DAML-S: Semantic Markup for Web Services. WWW resource for DAML-S version 0.9. May 5, 2003. On-line at: http://www.daml.org/services/daml-s/0.9/daml-s.html (2003)

[DMT04a] Distributed Management Task Force: DMI 2.0s Specification. WWW page. On-line at: http://www.dmtf.org/standards/dmi/spec (2004)

[DMT04b] Distributed Management Task Force: Web-Based Enterprise Management (WBEM) Initiative. WWW page. On-line at: http://www.dmtf.org/standards/wbem/ (2004)

[Esf01] Esfandiari, B. and Chandrasekharan, S., On How Agents Make Friends: Mechanisms for Trust Acquisition", Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies 2001, pp 27-34

[Esf98] Esfandiari, B., Deflandre, G. and Quinqueton, J., An Interface Agent for Network Supervision, IATA 1996 (Intelligent Agents for Telecom Applications), ed. S. Albayrak, IOS Press Publisher, 1998, pp 21-28

[Far02] Farrell, J. A., Kreger, H: Web Services Management Approaches. IBM Systems Journal, Vol. 41, No. 2. IBM On-line at: http://www.research.ibm.com/journal/sj/412/farrell.html (2002) 212-227

[Fen01] Feng, N., Gang, A., White, T., and Pagurek, B. (2001), Dynamic Evolution of Network Management Software by Software Hot-Swapping. In Proc. of the Seventh IFIP/IEEE International Symposium on Integrated Network Management (IM 2001), Seattle, May 14-18, 2001, pp. 63-76

[Fos02] Foster, I., Keselman, C., Nick, J. M., Tuecke, S.: Grid Services for Distributed Systems Integration. Computer, Vol. 35, No. 6 (June 2002). IEEE-CS (2002) 37-46

[Hon03] Hondo, M., Kaler, C. (eds.): Web Services Policy Framework (WSPolicy). May 28, 2003. BEA/IBM/Microsoft/SAP. On-line at: http://www-106.ibm.com/developerworks/library/ws-policy (2003)

[Kel03] Keller, A., Ludwig, H.: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. Journal of Network and Systems Management, Vol. 11, No 1 (Mar. 2003). Plenum Publishing (2003)

[Lam03] Lamanna, D.D., Skene, J., Emmerich, W.: SLAng: A Language for Defining Service Level Agreements. In Proc. of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems - FTDCS 2003 (Puerto Rico, May 2003). IEEE-CS (2003) 100-106

[Lay02] Laymann, F., Roller, D., Schmidt, M.-T.: Web Services and Business Process Management. IBM Systems Journal, Vol. 41, No.2. IBM (2002) 198-211

[Lud03] Ludwig, H., Keller, A., Dan, A., King, R.P., Franck, R.: Web Service Level Agreement (WSLA) Language Specification, Version 1.0, Revision wsla-2003/01/28. International Business Machines Corporation (IBM). On-line at: http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf (2003)

[Mac02] Machiraju, V., Sahai, A., van Moorsel, A.: Web Services Management Network: An Overlay Network for Federated Service Management. Research Report HPL-2002-234. Hewlett-Packard (HP) Laboratories Palo Alto. Aug. 21, 2002. On-line at: http://www.hpl.hp.com/techreports/2002/HPL-2002-234.pdf A shorter version published in Proc. of the Eight International Symposium on Integrated Network Management - IM 2003 (Colorado Springs, USA, Mar. 2003). IEEE (2003) 351-364

[Mar94] Marsh, S.: Formalising Trust as a Computational Concept, Ph.D. Thesis, Department of Computing Science and Mathematics, University of Stirling, Stirling, Scotland, (1994)

[Men01] Mennie, D., Pagurek, B.: A Runtime Composite Service Creation and Deployment Infrastructure and Its Applications in Internet Security, E-commerce, and Software Provisioning. In Proc. of the 25th Annual International Computer Software and Applications Conference – COMPSAC 2001 (Chicago, USA, October 2001). IEEE Computer Society Press (2001) 371-376$

[Ope04] The Open Group: Application Response Measurement – ARM. WWW page. On-line at: http://www.opengroup.org/tech/management/arm/ (2004)

[Pao03] Massimo Paolucci, Katia Sycara, Takuya Nishimura, and Naveen Srinivasan, "Using DAML-S for P2P Discovery," in Proceedings of the First International Conference on Web Services (ICWS'03), Las Vegas, Nevada, USA, June 2003, pp 203- 207

[Pel03] Peltz, C.: Web Services Orchestration and Choreography. Computer, Vol. 36, No. 10 (Oct. 2003) IEEE-CS (2003) 46-52

[Sah02a] Sahai, A., Durante, A., Machiraju, V.: Towards Automated SLA Management for Web Services. Research Report HPL-2001-310 (R.1). Hewlett-Packard (HP) Laboratories Palo Alto. July 26, 2002. $On-line at: http://www.hpl.hp.com/techreports/2001/HPL-2001-310R1.pdf (2002)

[Sah02b] Sahai, A., Machiraju, V., Sayal, M., van Moorsel, A., Casati, F.: Automated SLA Monitoring for Web Services. In Proc. of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2002 (Montreal, Canada, Oct. 2002). Lecture Notes in Computer Science (LNCS), No. 2506. Springer-Verlag (2002) 28-41

[ShA03] ShaikhAli, A., Rana, O. F., Al-Ali, R., Walker, D. W.: UDDIe: An Extended Registry for Web Services. In Proc. of 2003 Symposium on Applications and the Internet (SAINT'03) Workshops, Workshop on Services Oriented Computing: Models, Architectures and Applications (Jan. 2003, Orlando, USA). IEEE-CS (2003) 85-89

[Shr03] Sharma, A., Adarkar, H., Sengupta, S.: Managing QoS through Prioritization in Web Services. In Proc. of the 1st International Web Services Quality Workshop - WQW 2003 at the 4th International Conference on Web Information Systems Engineering - WISE 2003 (Dec. 2003, Rome, Italy) 20-28

[Ste95] Stevenson, D. W.: Network Management - What it is and what it isn't. Tutorial. On-line at: www.sce.carleton.ca/netmanage/NetMngmnt/NetMngmnt.html (1995)

[Sun04] Sun Microsystems: Java Management Extensions (JMX) - Documentation. WWW page. On-line at: http://java.sun.com/products/JavaManagement/reference/docs/index.html (2004)

[Tan01] Tan, L., Esfandiari, B., Pagurek, B. (2001), The SwapBox: A Test Container and a Framework for Hot-swappable JavaBeans. In Proc.of the WCOP (Workshop on Component-Oriented Programming) workshop (at ECOOP 2001), Budapest, Hungary, June 19, 2001

[Tos03] Tosic, V., Ma, W., Pagurek, B., Esfandiari, B.: On the Dynamic Manipulation of Classes of Service for XML Web Services. In Proc. of the 10th Hewlett-Packard Open View University Association (HP-OVUA) Workshop (Geneva, Switzerland, July 2003). Hewlett-Packard (2003)

[Tos04a] Tosic, V., Ma, W., Pagurek, B., Esfandiari, B.: Web Service Offerings Infrastructure (WSOI) – A Management Infrastructure for XML Web Services. In Proc. of NOMS (IEEE/IFIP Network Operations and Management Symposium) 2004, Seoul, South Korea, April 19-23, 2004. IEEE (2004)

[Tos04b] Tosic, V., Pagurek, B., Patel, B. Esfandiari, B., Ma, W.: Management Applications of the Web Service Offerings Language (WSOL). Accepted by Information Systems. Elsevier. Early version in: Proc. of CAiSE'03 (Velden, Austria,

June 2003). Lecture Notes in Computer Science (LNCS), No. 2681. Springer-Verlag (2003) 468-484

[Uta01] Utas, G., A Pattern Language of Feature Interactions, in: Rising, L., Design Patterns in Communication Software, Cambridge University Press, 2001

[Wei04] Weiss, M., Esfandiari, B., On Feature Interactions in Web Services and their Detection, to appear in Proceedings of ICWS 2004 (IEEE International Conference on Web Services)