

State of the Art: Service Composition

Steffen Higels, Dave Lewis
Knowledge and Data Engineering Group
Department of Computer Science

Abstract

This paper addresses the state of the art in service composition as driven by the current interest in web services. This paper examines this state of the art from the viewpoint of the application of state of the art to ubiquitous computing. It addresses the various issues related to service composition, including interoperability of service descriptions, service discovery, service brokering, security and trust and the modelling of service composition. Research directions in the application of service composition to ubiquitous computing is suggested.

Keywords:

Service composition, web services, WSDL, DAML-S

1 Introduction

Service oriented development involves developing well defined interfaces to units of software functionality and using these to integrate software into applications. As smart spaces will consist of software functionality from a very wide range of sources, a service oriented approach to smart space system development would seem appropriate. Service composition addresses techniques for rapidly integrating existing services into richer, composite services and therefore will be important in adaptive smart space systems.

2 Overview

Service Composition is the orchestration of a number of existing services to provide a richer composite service assembled to meet some user requirements. The current major interest in service composition, however, stems from the emergence of web services and the possibility of composing them to provide value-added services over the WWW. Service composition techniques typically involve expressing elemental services and composite services, the latter being compositions of elemental services and other composite services. The definition of composite services requires the expression of the flow of control and information between the elemental services. Techniques for this draw heavily on business process modelling and languages for workflow. The service composition domain also overlaps with software engineering in the assembly of systems from pre-existing software components. Architectural Description Languages (ADLs) address system assembly by assuming components offer well-defined services which are composed to meet system requirements. ADLs address static aspects of such composition, including the use of connectors to express the positioning of protocol or data transformation functions between services.

In the Smart Space environment users (or more likely their agents) will be faced with a changing array of local services, plus varying access to remote web-based services, as users move about. The task of orchestrating these services to meet the needs of whatever tasks the user currently wishes to undertake therefore requires adaptive service composition, i.e. the rapid composition and re-composition of services. The problem of how to dynamically compose Smart Space services with little a priori knowledge is therefore very analogous to the problem of web service composition, solution to which could thus be exploited for adaptive management of ubiquitous computing environments

3 Analysis

This analysis first categorises some of the different approaches that can be taken to service composition and then examines aspects of the state of the art in service interoperability, service discover, service brokering, security and trust and the modelling of service composition. These topics are of importance in service composition since they address the fundamental problems of combining services originating from a number of sources. Services require a common interoperability mechanism so that they can be readily interconnected in a single platform. They require mechanisms for registering available services so that service composition agents can later discover their availability. Service brokers extend service discovery to support the locating of services based on partial requirements, rather than a full service specification. Security and trust are essential both in communicating information between distributed services and in trusting that information submitted to a service will not be misused and that results are a true outcome of the advertised service. The modelling of services and service composition are the key to the level of automation that can be incorporated into service composition and these related activities.

3.1 Approaches to Service Composition

A useful taxonomy of composite service is defined in [chakraborty01b]. This differentiates between proactive composition, which is performed off-line for deployment on stable, always-up, resource rich platforms, and reactive composition where compound services are created on the fly under the auspices of some composition manager, often optimising for real-time parameters, e.g. available network bandwidth. The taxonomy also differentiates between mandatory-composite services which require the correct behaviour of all subcomponents, and optional-composite services where not all components need to be in place for service operation, e.g. data compression services optimising dataflow throughput. While the dynamic nature of smart spaces will require reactive composition, there will still be a need to integrate this with more stable model of a smart space host's business process model, which involves smart space administrators in proactive composition. This also raises the relatively unexplored issue of reconciling of personal business processes (of the Smart Space visitor) with organisation business processes (of the Smart Space host) [gary97a]. Smart Space environments, with their arbitrary mix of services and communication technologies will require reactive service composition that is optimal-composite in order to accommodate user movement between Smart Spaces, intermittent wireless connectivity and changes in accessibility to limited shared resources.

[chakraborty01b] also points out that many service composition system are centralised in how the flow of signals and information between services is managed, and this would seem to present a problem when applying them to Smart Space environments based on ad hoc networks and peer-to-peer applications, where no natural central control point may exist. However, [benatallah02a] provides a decentralised mechanism for controlling distributed control and information flow. This involves the

static analysis of state graphs that represent such flows to generate local flow rules that can be passed to lightweight schedulers operating with each service. This scheme is extended in [faudet01a] to support monitoring of the execution of composite services, another usually centralised function, in a decentralised, peer-to-peer manner.

3.2 Service Description and Interoperability

It is clear that service composition requires some level of commonality in the manner in which information is passed between services. Much of the interest in service composition in the WWW has been motivated by the emergence of the Web Services Description Language (WSDL) [wsdl] and its standardised binding to SOAP [soap] and, hence to HTTP, as a common communication infrastructure. As well as defining common communication protocol mapping, the transport of data as XML allow for greater flexibility in data encoding and transformation, e.g. using XSLT [xslt]. However, though WSDL would seem a likely service description mechanism for Smart Space services, it can potentially be bound to any communication protocol. In a Smart Space, where limited device profiles and specialised wireless networks may be common, service composition may have to operate over a variety communications protocols.

3.3 Service Discovery

Service discovery is the process of locating which services are available to take part in a service composition. The methods used for service discovery vary greatly depending on the environment in which the services exist. For a large scale local area network, a single database of online services might be appropriate. Service discovery is completed in this manner by the DySCo system [piccinelli01a], that uses a single service description manager, which keeps a database of all available services and the role they can play in a compound service.

As the scale of the network expands beyond that of a LAN, the methods used for service discovery must go beyond that of a simple “shout and reply” multicast. The core backbones of the internet would quickly be flooded by traffic if a sufficiently large number of clients asked every host on every network if they provide any services. For a huge array of services existing on a network on the scale of the Internet, a hierarchical lookup system like the domain name system [mockapetris87a] might be the best way to achieve scalability. The eFlow system developed by HPL addresses service discovery in this manner [casati00a]. If a service to complete the selected task cannot be found, the local service broker will ask any known external brokers if they know of the required service. For an ad-hoc wireless network, the problem becomes more complex and a further set of considerations must be made. The Anamika architecture [chakraborty01b], which focuses on service composition in an ad-hoc environment uses the service discovery features of Bluetooth to locate devices within range of the node requesting a service. In the event of a service not being located, every device that replies, is instructed to forward the request on to any devices they have within their range. This process is repeated recursively until the desired service is found.

Also worthy of note are frameworks like UPNP and JINI which represent a future direction for operating system network stacks. On top of the typical roles of such a stack (the basic delivery of information from an application on one computer to an application on another), they provide service

advertisement, service discovery and other network management protocols like NAT traversal to applications.

A UPNP enabled device uses a multicast (i.e. UDP based) version of HTTP to ask all devices on the network to respond if they are UPNP enabled as described in [Steinfeld 2001]. Recent versions of the Windows Operating System can be configured easily to become UPNP devices. Answers are transmitted by a unicast UDP variant of HTTP. Because service discovery is non-centralized, UPNP becomes impractical for any network larger than that of a large corporation. Each device can provide an XML list of all services it provides to any device.

JINI is a framework with similar goals to UPNP, though based heavily on Java. Service discovery happens via a Lookup Service, which is a centralized database of available services. If the hostname and port number of the Lookup Service are known beforehand, a simple unicast can be used to perform registration. It also supports multicast Lookup Service discovery. If a device requires a list of available services, it can locate the Lookup Service in a similar manner. If the network's router supports multicast routing, it can discover a Lookup Service on a remote network. The practicalities of this service discovery technique are explored further in [Chitrarasu et al. 1999].

3.4 Service Brokering

Service Brokering is the process of selecting which services should be included in any particular service composition. To make this choice, the service broker should be supplied with some self-description by the selected services. The criteria used to make this decision could consist of the time required by the service to complete execution, the role the service can play within a composition and the price charged by the service provider. When some pre-knowledge exists of what devices will be available in an environment, the selection of a service broker is relatively simple. It can simply be designated as such by the designer of the architecture. In a number of the service composition systems investigated, the task of service brokering was given to one or more predefined devices. In an ad-hoc network, where all devices might be of similar processing power and we have no guarantee that a given device will be available at a given time, some form of nomination and acceptance should be used. The Anamika architecture uses this approach. Another approach, taken by ICARIS [tosic00a], is to have a one centralized registration manager with which devices that are willing to act as service brokers can register themselves. The selection of which device will act as service broker is made by the registration manager.

3.5 Security and Trust

The secure communication between services is also a vital consideration in service composition for Smart Spaces. The ICARIS architecture approaches this problem in an interesting manner. Various cryptographic methods and protocols are encapsulated within different services, and can be chained together based on the needs of a given application. Secure, SSL style protocols (which combine asymmetric cryptography for initial key exchange and then symmetric cryptography for data exchange thereafter) can be built and then integrated with services and applications which were never designed with security in mind.

In an ad-hoc environment, it is assumed, we have many computers offering various services. For various reasons, it is important that a client (be it an end user or an adaptive service composer) can establish the identity of the machine providing the service. It is inevitable that in a truly ad-hoc environment, where we can never completely rely on any one device being present, that at some point, one device is going to have to make a "leap of faith" and agree to do business with a given service of which it has no prior knowledge. Taking this hindrance as a given, we can begin to examine an easy algorithm for establishing a degree of trust in this environment:

The client will determine that a given available service is capable of providing the functionality we require. Before trusting it with anything confidential, we establish what experiences other trusted devices in the environment have had and we ensure that the device is who it claims to be.

This is an ideal task for public/private key cryptography. Firstly, we generate a large random number. We send this to the service. The service then signs this number with its own private key and sends the signature back to us, along with its public key. We verify the signature using the public key. We broadcast to all devices on the network asking if they have also used a service running on this device. Along with the request, we send the service's public key, so each device can verify that they are referring to the same service. Each client's reply should contain two pieces of data:

1. A confirmation that the service in question, when used before, was using the supplied public key when used.
2. Some rating of the level of satisfaction with the service that was received.

How the results are handled is open to some research and debate.

There are two problems with this:

- Firstly, in the ad-hoc environment, we will probably have no pre-knowledge of the devices in the space. How do we trust them? How do we know that not only has the device providing the service hasn't been compromised, but that all the hosts telling us how great it is haven't also been compromised.
- Secondly, some thought needs to be put into what is being authenticated. Are we authenticating the device, along with all services that run on it, or do we authenticate the service only, and forget the notion of the device completely? Can we trust a service without trusting the underlying device, and how is this managed when service code is downloaded and run dynamically?

Mobile code security is also an important consideration in this context. In a dynamic computing environment, it is also important that the issues of trust are examined, to prevent a composition from being poisoned by a rogue service. Without the aid of hardware support, it is more or less impossible to trust completely any methods for verifying that a service running on a foreign system is what it claims to be[gong97a]. For this reason, we must be selective in the information that is transmitted, ensuring that sensitive data is transmitted only to a service in which we have a certain minimum level of trust.

One possible direction towards a practical solution is the wallet-of-certificates approach[lkagal01a], where trust is built up slowly through a series of cryptography based authentications. Following this thinking, a reasonable level of confidence in the identity of the service with which we are communicating can be developed.

3.6 Modelling Service Composition

Many different modelling languages have been used in expressing service composition, e.g. UML state diagrams in [benatallah02a], and often the semantics of these languages are derived from business process modelling techniques. The recent interest in web services and their expression in WSDL has resulted in XML based languages being used to define composition between WSDL service definitions, e.g. BPEL4WS, allowing service composition patterns to be readily exchanged between tools and thus reused. Such service composition specifications necessarily need to be expressed at the level of abstraction used for constituent services. Though techniques for developing business process models from business requirements are well established, they involved skilled manual activities and are not amenable to the dynamic service creation needed to meet the needs of Smart Space users lacking in business process modelling skills. A service composition transformation approach is taken to this problem in [wang00a], where existing user tasks are analysed as service compositions in one service environment and transformed to another when the user moves to another service environment. Though this effectively provides a form of seamless session mobility between smart spaces, it does not help users in composing services to achieve new tasks, perhaps taking advantage of novel services found in a recently entered Smart Space. A ‘semantic gap’ therefore exists between the model the user possesses of what they want to do, and the service composition models that simply express how this may be accomplished. Some work has been performed in reconciling the behaviour of a system with a high level representation of its requirements [feather98a]. However, even these high level requirements are complex to express and are typically elicited by skilled requirements engineers. Adaptive techniques are used in other contexts, such as e-learning, to dynamically map basic user profile information to adapt hypermedia documents to user’s knowledge and device display capabilities [conlan02a]. A close analogy between the expression of composition in hypermedia authoring languages and ADLs is identified in [muchaluat-saade01a]. As ADLs address many of the static aspects of service composition, this may point to the utility of adaptive hypermedia techniques in adaptive service composition.

Ultimately, however, accurate mapping of user level requirements to service composition requires modelling of the real world which forms the context in which the user naturally expresses their Smart Space usage needs. Ontologies provide a way of capturing and exchanging models of the real world and making them available to automated agents. The DARPA Agent Mark-up Language initiative is defining XML-based standards for supporting the development of the Semantic Web [berners-lee01a], which aims to make the content of the web more amenable to processing by automated agents. As part of this initiative DAML-S [damls02a] uses the DAML+OIL ontology language to provide semantic mark-up of web services. This includes modelling the links between a service and the outside world, by using ontologies for expressing the inputs, outputs, preconditions and effects of a service and the resource which provides it, in a way that can be mapped to semantic descriptions of the real world. DAML-S uses WSDL to ground its service definitions and thus bind it to the actual interface technology used for a service instance. DAML-S also provides mechanisms for defining composite services. Thus the use of WSDL for defining all Smart Space services, coupled with the integrated semantic richness added by DAML-S, opens the door to using a range of rule-based, inference and other AI techniques to the problem of adapting user needs to Smart Space service composition [mcilraith01a]. Currently such application range from using of rule-engines reasoning on simple conditions on service input and output to perform limited automated service composition [ponnekanti02a] to the use of situation calculus to reason about semantic information in DAML-S

specs in tools to aid developers in the simulation, verification and automated composition of web services [narayanan02a].

It is interesting to note the web services are not the first domain to encompass machine reasoning about service composition. Machine reasoning has previously seen application to service engineering, principally in the telecommunications industry. Formal languages such as the Service Description Language (SDL) [ellsberger97a] were used to provide formal definition of services implemented in component-oriented telecommunication control systems, often referred to as intelligent networks. Reasoning with such formal service descriptions was used initially to detect unwanted feature interactions in the service creation process for intelligent networks, but research was also conducted into using them for user evaluation trials, design simulations and automation of test case generation [lodge99a]. SDL bears many similarities to DAML-S, as both express logical assertions about preconditions and post-conditions, i.e. effects, of service invocation, and can also express logical relationships and constraints between service inputs and output. However, the use of formal reasoning in intelligent network service creation was often seen as an addition to the service creation process, and therefore an overhead the benefits of which had to be directly assessed against the equivalent costs of traditional software engineering processes. Also, the logical concepts used in defining formal assertions for a particular intelligent network service was tied to that particular service and offered little benefit outside the development of that service. At best such formally defined objects were restricted to reuse in the intelligent network software market which was dominated by a few large players, and thus not open to the utility of formal definitions as a commodity or shared resource.

4 Research Directions

In identifying research directions related to service composition, a distinction needs to be made between research into algorithms for defining optimal service compositions, and the assembly of a suitable service platform on which different such algorithms can be evaluated. In terms of a service platform, considerations are:

- WSDL would seem to provide a good basis for the definition of service syntax and forms the basis for a lot of web service composition research. The ability to bind WSDL descriptions to various protocols, allows application specific services to be implemented or simulated on one communication platform while remaining open to use on other development platforms. A range of WSDL tools are available.
- Consistent with this choice, the use of DAML-S for defining the semantic of web services may provide a common platform for a number of service composition approaches. The movement from the use of DAML+OIL to the derivative OWL which is being standardised by the W3C points to DAML-S as a likely basis for future standardisation of semantic web definitions.

In terms of specific research areas related to service composition, the following should be considered:

- Automated Service Composition: Smart Spaces require dynamic and thus largely automated service composition. Mechanisms for such zero-code composition are required [kiciman01a]. We aim to examine the extent to which service composition may be automated and will examine schemes where pre-existing, manually developed compositions can be integrated with and used to improve the automated service composition process.
- Bridging the semantic gap: Though DAML-S may provide us with a way of openly exchanging semantic information on services, exactly how this is used to map service to what the user

wants to do is an open issue. This involves both matching semantics between service-oriented models of what user's wish to do, e.g. as tasks, and service composition models based on existing services and they semantics they possess.

- **Ontologies for Smart Space Management:** Can we establish some common ontologies for adaptive smart space management? This will examining the possible application of existing management models in an ontological form to the management of smart spaces.

5 References

[benatallah02a] Benatallah, B., Dumas, M., Sheng, Q.Z., Ngu, A.H.H. (2002), 'Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services', Proceedings of the 18th International Conference on Data Engineering (ICDE'02), ISBN: 0-7695-1531-2, March 2002, pp 297 – 308

[berners-lee01a] Berners-Lee, T., Hendler, K., Lassila, O. (2001), 'The Semantic Web', Scientific American, pp 35-43, Issue 284 (3), 17th May 2001

[casati00a] Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M. (2000), 'Adaptive and Dynamic Service Composition in eFlow', HPL-2000-39 20000406 External, March 2000

[chakraborty01b] Chakraborty, D., Joshi, A. (2001), 'Dynamic Service Composition: State-of-the-Art and Research Directions', Technical Report TR-CS-01-19, Department of Computer Science and Electrical Engineering, University of Maryland, 19th December 2001

[chakraborty02a] Chakraborty, D., Perich, F., Joshi, A., Finin, T., Yesha, Y. (2002), 'A Reactive Service Composition Architecture for Pervasive Computing Environments', Technical Report TR-CS-02-03, Computer Science and Electrical Engineering, University of Maryland Baltimore County

[chitrarasu99] Chitrarasu, M., Joseph, K., Rao, M., (1999) "Jini by Example – Whitepaper", published online.

[conlan02a] Conlan, O., Wade, V., Bruen, C., Gargan, M. (2002), 'Multi-Model, Metadata Driven Approach to Adaptive Hypermedia Services for Personalized eLearning', Proceedings of Second Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Eds. De Bra P., Brusilovsky, P., Conejo, R., Springer, LNCS 2347, May 2002, pp 100-111

[damls02a] 'DAML-S: Semantic Markup for Web Services', The DAML Service Coalition, <http://www.daml.org/services/>, October 2002.

[ellsberger97a] Ellsberger, J., Hogrefe, D., Sarma, A. (1997), "SDL, Formal Object-oriented language for communicating systems", Prentice-Hall

[fauvet01a] Fauvet, M.C. Dumas, M., Benatallah, B., Paik, H.Y. (2001), 'Peer-to-Peer Traced Execution of Composite Services', Second International Workshop on Technologies for E-

Services, Eds. Fabio Casati; Dimitrios Georgakopoulos; Ming-Chien Shan, Rome Italy, 14-15 Sep. 2001, Springer, Heidelberg, Germany, pp103-117

[feather98a] Feather, M.S., Fickas, S., van Lamsweerde, A., Ponsard, C. (1998), 'Reconciling System Requirements and Runtime Behaviour', Proceedings of Ninth International Workshop on Software Specification and Design, IEEE, 16-18 April 1998, pp 50 – 59

[gary97a] Gary, K., Lindquist, T., Koehnemann, H., Sauer, L. (1997), 'Automated Process Support for Organizational and Personal Processes', Proceeding of the International ACM SIGGROUP conference on supporting group work, Phoenix, Arizona, ACM Press, pp 221-230

[gong97a] - Li Gong (1997), 'Survivable Code is Hard to Build', DARPA Workshop on Foundations for Secure Mobile Code, March 1997

[kagal01a] - Lalana Kagal, Tim Finin, Anupam Joshi (2001), 'Moving from Security to Distributed Trust in Ubiquitous Computing Environments', unpublished [kiciman01a] Kiciman, E., Melloul, L., Fox, A. (2001), 'Position Summary: Towards Zero-Code Service Composition', Eighth Workshop on Hot Topics in Operating Systems May 20 - 22, 2001 p.0172

[lodge99a] Lodge, F., Kimbler, K., Hubert, M. (1999) "Alignment of the TOSCA and SCREEN Approaches to Service Creation" in Proceedings of the 6th International Conference on Intelligence in Services and Networks, Barcelona, Spain, Springer-Verlag, pp277-290

[mcilraith01a] McIlraith, S.A., Son, T.C., Honglei Zeng, H. (2001), 'Semantic Web Services', IEEE Intelligent Systems, 16(2), March/April 2001

[muchaluat-saade01a] Muchaluat-Saade, D.C., Soares, L.F.G. (2001), 'Towards the Convergence between Hypermedia Authoring Languages and Architecture Description Languages', Proceeding of the ACM Symposium on Document Engineering, Atlanta, Georgia, USA, ACM Press, pp 48-57

[narayanan02a] Narayanan, S., McIlraith, S.A., (2002) "Simulation, Verification and Automated Composition of Web Services", Proceedings of 11th World Wide Web Conference, May 7-11, 2002, Honolulu, Hawaii, pp 77-88

[piccinelli01a] Piccinelli, G., Mokrushin, L. (2001), 'Dynamic e-service composition in DySCo', 21st International Conference on Distributed Computing Systems Workshops (ICDCSW '01) April 16 - 19, Mesa, Arizona p. 0088

[ponnekanti00a] Ponnekanti, S.R., Fox, A. (2002), "SWORD: A Developer Toolkit for Web Service Composition", To appear in The Eleventh World Wide Web Conference (Web Engineering Track), Honolulu, Hawaii, May 7-11, 2002 (<http://swig.stanford.edu/public/publications>)

[soap] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Frystyk Nielsen, H., Thatte, S., Winer, D. (2000), "Simple Object Access Protocol (SOAP) 1.1", W3C Note 08 May 2000, <http://www.w3.org/TR/SOAP>

[steinfeld] Steinfeld, E., (2001) “Devices that play together, work together”, EDN Magazine September 2001

[tosic00a] Tasic, V., Mennie, D., Pagurek, B. (2000), ‘On Dynamic Service Composition and Its Applicability to E-Business Software Systems’, Carleton University, Ottawa, Canada Published online

[uddi00] “UDDI Technical White Paper”, 6th September 2000, www.uddi.org

[wang00a] Wang, Z., Garlan, D. (2000), ‘Task Driven Computing’, CMU document CMU-CS-00-154, May 2000.

[wsdl] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S. (2002), “Web Services Description Language (WSDL) 1.1”, W3C Note 15 March 2001, <http://www.w3.org/TR/wsdl>

[xslt] XSL Transformations v1.0, W3C 1999: <http://www.w3.org/TR/xslt>