

# Comparative Evaluation of Converged Service-Oriented Architectures

Davy Preuveneers, Julien Pauty, Dimitri Van Landuyt,  
Yolande Berbers, Wouter Joosen  
Department of Computer Science  
Katholieke Universiteit Leuven  
Celestijnenlaan 200A, Leuven, Belgium  
{firstname.lastname}@cs.kuleuven.be

## Abstract

*In this paper we evaluate and compare five commercial off-the-shelf platforms for building converged service-oriented architectures in an ambient computing environment. To this end, we have identified several categories of requirements that support the always-connected digital lifestyle of the mobile user. Key requirements are providing convergence – joining the worlds of web services and telecom services – and offering context awareness in order to tailor services better to the user’s environment, preferences and needs. Our evaluation shows that currently none of the investigated architectures completely fulfills the two aforementioned requirements in order to deploy a variety of future converged mobile and context-aware services. However, the resulting comparison can be used as a practical guideline for decision-makers to select a platform to build a service-oriented architecture adjusted to the specific requirements within their application domain.*

## 1 Introduction

A service-oriented architecture (SOA) represents the current state of the art in software architecture for the rapid deployment of new services. It enables the creation of new services or applications by connecting together existing services and proposes functions to manage the service lifecycle, such as the deployment and updating of services. The main motivation behind SOA for a company is to create a business-aligned architecture to better react to changing customers’ needs. If the market changes, new services can be created by reusing existing services and by developing new services, where needed. An SOA also enables a company to leverage previous infrastructure investments, by exposing legacy services as traditional services in the architecture: an SOA increases code reuse and modularity.

The goal of this paper is to paint an accurate and up-to-date picture of the current state of the art in service-orientation by comparing the most representative commercial solutions in this field: the BEA WebLogic SIP Server [3], the jNetX Open Convergent Feature Server [10], the IBM Service Provider Delivery Environment [7], the Cape Clear Enterprise Service Bus [4], and the Microsoft Connected Services Framework [12]. The criteria for inclusion were twofold: (1) factors such as commercial relevancy, availability of technical information, use of a standardized component model, and many more; (2) adequate and sufficiently wide overview of the entire spectrum of established service-oriented solutions. The motivation of this study is to assist the service provider to choose the best platform for building service-oriented architectures for offering converged web services and telecom services adapted to the always-connected digital and mobile lifestyle of the future.

The evaluation of these commercial off-the-shelf platforms was carried out in terms of several key requirements that were identified in a concrete real-life usage scenario in the targeted advertising and publishing domain. In summary, this scenario is as follows: Instead of building a standard news website, a news publisher decides to base his business on the service-oriented paradigm. This allows him to offer the same service *on a wide variety of devices* (web server, cellphone, TV, giant billboard in the city), and more *flexibility to manage and integrate* with other services. The news service is combined with a service that allows readers to enrich news articles with comments or new material, or to start an online discussion with others. Targeted publishing is made possible by showing advertisements based on the very concrete location or *context* of the user.

In Section 2 we provide a brief overview of the evaluated service platforms and describe how they are used. Section 3 discusses the requirements that an SOA must fulfill to deliver converged services to mobile users. We evaluate and compare the architectures in Section 4 before concluding in Section 5.

## 2 Main characteristics of the architectures

This section briefly describes the service architectures under investigation in this paper. The list of architectures is not meant to be exhaustive. Instead, we wanted to include architectures relying on different technologies (such as JAIN SLEE, SIP servlets, HTTP servlets, web services, Enterprise Java Beans) in order to show the different approaches currently followed in the service oriented world.

**The BEA WebLogic SIP Server:** With the BEA WebLogic SIP Server [3], developers can deploy HTTP servlets, SIP servlets, EJBs on a single J2EE platform. Telecommunication services are executed inside the SIP servlet container, IT services in the HTTP servlet or EJB container.

**The jNetX Open Convergent Feature Server:** The jNetX OCFS [10] is a service architecture implementing the JAIN SLEE specification [14, 1]. JAIN SLEE is the abbreviation for the Java APIs for Integrated Networks Service Logic Execution Environment, a high throughput, low latency event processing application environment.

**The IBM Service Provider Delivery Environment:** IBM SPDE [7] combines the IBM WebSphere MQ Integrator (a message broker that routes and transforms messages coming from one service in order to adapt them to another service) and the IBM WebSphere Everyplace Server for Telecom, which corresponds to the well known IBM WebSphere J2EE application server and a telecommunication toolkit to access telecommunication services.

**The Cape Clear Enterprise Service Bus:** The architecture provided by Cape Clear [4] is based on web services and open standards. This architecture relies on the Eclipse development environment for service creation and provides a tool to model business process graphically for non-technical users to create new services.

**The Microsoft Connected Services Framework:** Microsoft CSF [12] is a service framework relying on the .NET technology. It enables the development of composite web services and manages common functions of service control and collaboration such as session management, identity management, service discovery and profile management.

As an evaluation copy could not be obtained for all of these service architectures and platforms, the comparison presented in the following section was limited in some respect to a qualitative assessment based on the specifications and other documentation made available by the providers.

## 3 Requirements

The growing presence of mobile devices, such as laptops, PDAs and smartphones, along with advances in wireless network communication technologies, promises to

change drastically the human-computer interaction. Mobile and ubiquitous computing are new emerging computing paradigms that change the way applications are designed, implemented and consumed. They create new opportunities for making the applications more intelligent and supportive to the user in a service-oriented world.

Therefore, we identified several requirements with a setting as outlined in the usage scenario in mind and compare five service architectures covering the telecommunication and web service domains with respect to these requirements. They are grouped in the following categories: convergence between IT and telecom, service management, context awareness, compliance with industry standards (e.g. w.r.t. identity management), and non-functional requirements such as scalability and availability.

### 3.1 Converged service architecture

A converged architecture enables the creation of services that emerge through the combination of telecommunication services, such as VoIP conversations, and web services, such as online bookstores. The success of a converged architecture can be measured by its support for interoperability. Interoperability between IT and telecommunication-oriented architectures is a complex issue. Both propose similar though domain specific standards to which their services adhere in order to ensure interoperability within their respective domains. Another issue is the difference in architectural style of these service-oriented architectures. An IT web service is essentially transaction-based and relies on an architecture that often builds upon the functionality of an enterprise service bus to provide message brokering, routing, data translation and transformation. A telecommunication service architecture needs to deal with a multitude of point-to-point connections and short-lived events that must be generated, propagated and processed with a low latency [13] to guarantee a minimal quality of service level. Integrating both kinds of service is hard without sacrificing the performance of the telecommunication services.

We distinguish two kinds of bridging mechanisms: (1) *loose coupling* between IT and telecommunication services, such as a gateway (i.e. a one-way adapter which exposes a telecommunication service as a web service, or a web service as a telecommunication service); (2) mechanisms which create a *tight coupling* between IT and telecommunication services. The first bridging mechanism complies with service orientation.

**Scenario:** The always-connected digital lifestyle of the mobile user demands new ways for service provision. Next generation communication networks enable services that go beyond wireless and wireline voice communication. A successful news provider will deliver on-demand content and services by both mobile and fixed access.

**Discussion and evaluation:** Some of the architectures under investigation provide a loosely coupled bridging mechanism by means of an OSA/Parlay gateway. An OSA/Parlay gateway exposes telecommunication services as traditional web services. Despite being service-oriented, such a bridging mechanism is often limited because only those functionalities of telecommunication services are exposed whose semantics are compatible with those of web services, typically configuration functionalities. IBM SPDE and Microsoft CSF integrate an OSA/Parlay gateway in their architecture. BEA WebLogic SIP Server and jNetX/OCSF propose a bridging mechanism that tightly couples IT and telecommunication services. jNetX/OCSF is a JAIN SLEE platform, which proposes a bridging mechanism with a J2EE container that enables an Enterprise Java Bean (EJB) to trigger an event in the JAIN SLEE container and that also allows a JAIN SLEE component to invoke an EJB. Such a bridging is done at the Java code level, creating a strong coupling between services. BEA WebLogic SIP Server is an HTTP and SIP servlets container. BEA's bridging mechanism relies on a shared session mechanism, meaning that the internals of services can be exposed to other services via this shared session. The Cape Clear ESB does not provide any mechanism for bridging the gap between IT and telecom. The following table presents the general ranking scale for this requirement:

Ranking	Description
++	Bridging mechanism that complies to service orientation
+	Bridging mechanism that does not comply with service orientation
0	No bridging mechanism

### 3.2 Support for service life-cycle management

Life-cycle management is a mandatory feature for any business-aligned service architecture. This requirement involves support for the deployment and updating of services (offline and online) of services, service dependency management, service dispensal, maintenance and testing. As most of these classic life-cycle management requirements are fulfilled by every platform that supports service-oriented architectures, and as time to market and a quick response to changing needs are crucial success factors, we specifically focus on the ease of service management and creation by composition of converged services. Indeed, a service-oriented architecture enables the creation of a service by composing existing services. SOA providers have created graphical tools that enable non-technical users to compose and adapt services. Such tools should enable companies to create services more quickly while reusing existing services and assets. In order to evaluate the architecture from the ser-

vice creation point of view, we categorize service managers based on their level of technical expertise:

- **Non-experts:** non-experts do not have any kind of programming experience or telecommunication knowledge, but they can design services (e.g. graphically) and test them;
- **Integrators:** integrators primarily participate in development related to the integration with other elements, as well as in network emulation testing;
- **Developers:** professional developers that have advanced programming skills.

**Scenario:** A new publisher should not be familiar with telecommunication systems to deliver news articles to clients. Neither should he require programming skills to add new services. A simple click on the button should be enough to link a poll service to a news article service.

**Discussion and evaluation:** Among the contenders, only IBM SPDE and Cape Clear explicitly provide tools for non-technical people to graphically create and design services. The graphical tools help to model business processes, which do not require high-level programming skills at all. jNETx also offers graphical tools, in the form of a variety of graphical workspaces, reusable components, and tools for all the steps in the service delivery process (design, development, testing and integration). Despite being graphical, these tools clearly require telecommunication knowledge and programming skills to create an application. The BEA WebLogic SIP Server and Microsoft CSF architectures only provide tools for experienced programmers.

Ranking	Description
++	Full life-cycle management, minimal technical expertise
+	Full life-cycle management, highly trained professional
0	Minimal life-cycle management without support for service creation or composition

### 3.3 Context awareness

Context awareness is particularly important to propose relevant services to mobile users. By storing and analyzing context data, such as the user's location and his preferences, services can be selected and adapted according to the current situation of the user. Therefore, the architecture must provide mechanisms to capture and store user context, either as part of the architecture or as an enabling service. Indeed, context data can be extracted from different sources such as the network infrastructure (location, cell-id), from the user terminal (battery level, preferences), from a presence server (online, busy, away) [15]. The architecture must also make this context available to service developers, respecting any existing privacy regulations.

**Scenario:** Service provision can be localized, personalized to the preferences of the user or adapted to the capabilities of the end-user device. Dealing with this contextual information is key to deliver services in a mobile network environment to a heterogeneous population.

**Discussion and evaluation:** In general, the architectures we investigated have little support for context awareness. Most of them support presence, location and user profiles, except Cape Clear which has no support for context awareness at all. However, none of them supports context reasoning [16, 20], which is an essential feature to go beyond pure location- or presence-based services. Context reasoning is particularly important in the service domain, because the way services may use various kinds of derived context data is not known in advance.

Ranking	Description
++	Advanced context-aware system, including context reasoning, distribution and sharing
+	Basic context awareness (location and/or user profile)
0	No context awareness

### 3.4 Federated identity management and policy enforcement

A service architecture targeting mobile users typically has a large number of concurrent users. The management of multiple versions of user identities across multiple services makes policy enforcement, without proper user administration and identity management, a daunting task. For this reason, the architecture must provide access management software (user management together with authentication techniques) in order to centrally control user access and enable single sign-on (SSO) through a policy server that grants authorization rights to each application.

Policy enforcement is a way to check an individual, a computing system or another service before allowing access to a particular end-user (business) service. Policies may define access restrictions (access control policies), privacy guarantees, billing mechanisms or quality of service provision. The architecture must be able to enforce these policies at runtime. For example, single sign-on through a policy server may grant end-users authorization rights to each service deployed on the architecture without the user going through a cumbersome login/password authentication process for each application independently. Policies of interest to a converged service architecture for mobile users include:

- User authentication and access control (authorization) to services;
- Privacy policies, for services that need access to private user information and sensitive context data;

- Service Level Agreements (SLAs), guaranteed quality of service provision, e.g. for communication services;
- Real-time billing of service use.

**Scenario:** For customized service delivery, the identity of the client plays an important role. Policies and federated identity management help the news provider to centralize user management and personalize content across all its news services. Single sign-on is particularly interesting for mobile services, because the user terminal has limited interaction capabilities, which can make the authentication even more tedious.

**Discussion and evaluation:** For this requirement we do not define a ranking scale, but we list for each architecture the relevant supported standards and the federated identity management system that is used. Many architectures comply to industry standards by using the WS-Policy [8], WS-Federation [6] and derivative web service standards, as well as the Liberty [11] and other single sign-on standards.

### 3.5 Non-functional requirements

Since the targeted service-oriented architecture is to be accessed by a large number of users, *scalability* is a key requirement for the architecture. Some aspects related to scalability of an SOA include the management of Service Level Agreements, load-balancing of services and dynamic routing of service requests.

*Performance* monitoring is crucial in an SOA. Previous work [5] has shown that some SOA implementations rely on large (in the order of megabytes) and complex XML message formats causing insufficient throughput, terrible performance and scalability results. Since an SOA is a set of very loosely coupled and reusable components, it is important to trace interactions when degradation of performance for one service break other services within the SOA.

As most services are transported over HTTP, reliability in a service-oriented architecture includes a.o. *reliable messaging*. Reliable message delivery means the ability to ensure delivery of a message with the desired level of quality of service: (1) Message sent at least once (guaranteed delivery), (2) Message sent at most once (guaranteed duplicate elimination) or (3) Message sent exactly once (guaranteed delivery and duplicate elimination). There are several specifications that are supposed to address this: SOA-Reliability for HTTP [21], HTTPR [18], Web Services Reliable Messaging [2], Web Services Reliability [9].

**Scenario:** Quality of Service (QoS) is a key concern for all service providers. A news provider must ensure that the client gets what he paid for. Scalability to thousands of users and guaranteed message delivery without delays will be crucial for the success of the news publisher.

Service architecture	Converged services	Life-cycle management	Context-awareness	Standard compliance	Non-functional requirements
BEA WebLogic SIP Server	+	+	+	SLA, Billing, QoS	Clustering, Load balancing
jNetX/OCFS	+	+	+	Billing	Vertical and horizontal scalability
Cape Clear ESB	0	++	0	WS-Policy, SSO	Load balancing and high availability
IBM SPDE	++	++	+	WS-Policy, WS-Federation, Liberty	Load balancing and replication
Microsoft CSF	++	+	+	Privacy, Billing, QoS, Biztalk SSO, WS-Federation	Clustering, FCAPS

**Table 1. Comparative evaluation of commercial off-the-shelf service-oriented architectures**

**Discussion and evaluation:** Most of today’s SOA projects are depending on an enterprise service bus (ESB) to provide message reliability, exception handling, and publication-subscription model capabilities. FCAPS (fault-management, configuration, accounting, performance, and security) is the ISO Telecommunications Management Network model and framework for network management. Vertical scalability refers to the fact that a node in the system can be upgraded by adding more resources to process more transactions. Horizontal scalability means that more nodes can be added to the system to better handle the growing amount of transactions. For example, the JAIN SLEE specification has been designed with scalability and performance in mind:

- **Vertical:** Vertical scalability enables parallel service logic execution utilizing SLEE multi-threading that allows splitting processing over different CPU units.
- **Horizontal:** Distributing software horizontally over multiple hosts ensures that a single fault cannot bring the whole system down. It provides network and gateway load balancing.

For these requirements, we do not provide a ranking scale. Indeed, evaluation of these requirements is usually very difficult, because evaluating these architectures empirically is practically impossible. Therefore, for each architecture we provide a summary of the main points for these requirements as outlined in the available documentation.

## 4 Evaluation

In the second section of this paper we have introduced different requirements for an architecture for mobile services. In the preceding section, we have studied different architectures with respect to these requirements. In this section, we provide a side-by-side comparison of these architectures with respect to these requirements. Table 1 presents the evaluation.

We can place several side remarks next to this table. Firstly, only IBM SPDE and Microsoft CSF provide a bridging mechanism that complies with service orientation. Nevertheless, this mechanism is limited to telecommunication services that are semantically compatible with web services. Secondly, life-cycle management is well implemented in these architectures, with a majority of architectures providing functionality for updating services at runtime. Thirdly, the WS-Policy is the standard that seems to be adopted by the majority of the architectures that are based on web services, e.g. IBM SPDE, Cape Clear.

Although none of the services architectures under investigation in this paper received the lowest score for life-cycle management, we have evaluated other service architectures with only very limited support in this area. Due to the limited scope and intentions of some of these architectures, comparison with other full-fledged architectures would be unfair. Also due to space constraints, we have left out a description of the capabilities of these architectures in the previous discussions and in the final evaluation.

An architecture for mobile services needs a bridging mechanism to leverage IT and telecommunication services, and a context-aware system to propose relevant services to mobile users. If we look at the table, no single architecture provides a bridging mechanism which is compliant to service orientation, *and* an advanced context awareness system. The field where these commercial systems lack most support is context awareness: often only very basic context awareness functionality is introduced. This suggests that interesting research could be done to create such an architecture. One architecture of interest from the research community in this area is the one provided by the IST Amigo [17] project providing research in Ambient Intelligence for the networked home environment. Though it lacks support for converged services, it has developed an open, standardized, interoperable middleware and attractive context-aware user services. It also supports interoperability between equip-

ment and services within the networked home environment by using standard technology when possible.

The evaluated architectures are rather different. Therefore, we cannot use this table to compare the architectures in order to claim that one particular architecture is the best for all purposes. However, the table can be used to compare similar architectures, particular features or the standards compliance of SOA providers such as IBM SPDE and Microsoft CSF. For example, according to the evaluation table, we can state that IBM SPDE is more advanced than Microsoft CSF in terms of support for full life-cycle management. Furthermore, these architectures are not always seen as direct competitors. However we presented them independently, in practice, architectures providing different functionalities can usually be combined. An example is the BEA WebLogic SIP Server with a JAIN SLEE server, or IBM SPDE with Ubiquity A/S [19], another SIP-based application server.

## 5 Conclusion

In this paper we have studied and evaluated several commercial platforms for building off-the-shelf service-oriented architectures for converged services. We have chosen the architectures outlined in this paper so that most of the current approaches for service architecture in the IT and telecommunications world have been presented. We have discussed architectures based on SIP servlets, JAIN SLEE, HTTP servlets, EJB and Web Services. To evaluate the architectures we have identified several categories of requirements: (1) convergence between IT and telecom; (2) service management; (3) context awareness; (4) compliance with industry standards for policy enforcement and federated identity management; and (5) support for varying non-functionals, such as availability and scalability. Two major requirements for a service architecture that are necessary to support the always-connected digital lifestyle of the mobile user are “convergence” and “context awareness”. According to the evaluation, currently no architecture optimally fulfills these two requirements. Our future work will be dedicated to the study, implementation and integration of such support in a new or existing architecture.

## References

- [1] B. Van Den Bossche, et al. Evaluation of Current Java Technologies for Telecom Backend Platform Design. In *International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pages 699–709, July 2005.
- [2] BEA Systems, IBM, Microsoft and TIBCO Software. Web Services Reliable Messaging Protocol (WS-ReliableMessaging). <http://xml.coverpages.org/WS-ReliableMessaging200502.pdf>, February 2005.
- [3] BEA Systems, Inc. Beas WebLogic SIP Server Whitepaper. [http://www.bea.com/content/news\\_events/white\\_papers/BEA\\_WL\\_SIP\\_Server\\_ds.pdf](http://www.bea.com/content/news_events/white_papers/BEA_WL_SIP_Server_ds.pdf), 2005.
- [4] Cape Clear. Service-Oriented Architecture (SOA). <http://www.capeclear.com/technology/architecture/>, 2006.
- [5] F. Cohen. FastSOA: Accelerate SOA with XML, XQuery, and native XML database technology. The role of a mid-tier SOA cache architecture. <http://www-128.ibm.com/developerworks/xml/library/x-accsoa/>, February 2006.
- [6] IBM. Web Services Federation Language - Federated Identity and Security. <ftp://www6.software.ibm.com/software/developer/library/ws-fed.pdf>, July 2003.
- [7] IBM. Service-oriented Architecture. <http://www-306.ibm.com/software/info/openenvironment/soa/>, 2006.
- [8] IBM, BEA Systems, Microsoft, SAP AG, Sonic Software and VeriSign. <http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-polfram/ws-policy-2006-03-01.pdf>.
- [9] K. Iwasa, J. Durand, T. Rutt, M. Peel, S. Kunisetty, and D. Bunting. WS-Reliability v1.1. <http://docs.oasis-open.org/wsrn/ws-reliability/v1.1/wsrn-ws-reliability-1.1-spec-os.pdf>, November 2004.
- [10] jNetX. jNetX OCFS product overview. <http://jnetx.com/index.php?id=370>, 2006.
- [11] Liberty Alliance Project. Digital Identity Defined. <http://www.projectliberty.org/index.php>, 2004.
- [12] Microsoft. Microsoft Connected Services Framework. <http://www.microsoft.com/serviceproviders/solutions/connectedservicesframework.mspx>.
- [13] P. O’Doherty. JAIN SLEE Performance Analysis. [http://java.sun.com/products/jain/JSLEE\\_Performance\\_Analysis.html](http://java.sun.com/products/jain/JSLEE_Performance_Analysis.html), 2003.
- [14] P. O’Doherty. JAIN SLEE Principles. [http://java.sun.com/products/jain/article\\_slee\\_principles.html](http://java.sun.com/products/jain/article_slee_principles.html), 2003.
- [15] SIMPLE WG. SIP for Instant Messaging and Presence Leveraging Extensions. <http://www.ietf.org/html.charters/simple-charter.html>, March 2006.
- [16] T. Strang., et al. CoOL: A Context Ontology Language to enable Contextual Interoperability. In *LNC3 2893: Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003)*, volume 2893 of *LNC3*, pages 236–247. Springer, November 2003.
- [17] The Amigo project. <http://www.hitech-projects.com/euprojects/amigo/>, 2006.
- [18] S. Todd, F. Parr, and M. Conner. A Primer for HTTP - An overview of the reliable HTTP protocol. <http://www-128.ibm.com/developerworks/webservices/library/ws-phhttp/>, March 2005.
- [19] Ubiquity. Ubiquity SIP Application Server. [http://www.ubiquitysoftware.com/products/SIP\\_Application\\_Server.php](http://www.ubiquitysoftware.com/products/SIP_Application_Server.php).
- [20] X. Wang, T. Gu, D. Q. Zhang, and H. K. Pung. Ontology-Based Context Modeling and Reasoning using OWL. In *Context Modeling and Reasoning Workshop (CoMoRea’04)*, 2004.
- [21] Y. Goland. SOA-Reliability (SOA-Rity) for HTTP. <http://www.goland.org/draft-goland-http-reliability-00.html>, 2005.