



Contents lists available at ScienceDirect

# Simulation Modelling Practice and Theory

journal homepage: [www.elsevier.com/locate/simpat](http://www.elsevier.com/locate/simpat)

## Editorial

# Towards dependable service-orientated computing systems

## 1. Service-orienting computing systems

Service-orienting computing systems refer to computing and information systems that are developed in service-oriented architecture, service-oriented computing paradigm, and service-oriented development lifecycle.

The development of service-oriented computing systems is completing the first decade in its lifespan. Service-oriented computing systems started with the concept of Service-Oriented Architecture (SOA). As distributed architecture, SOA considers a software system consisting of a collection of loosely coupled services that communicate with each other through standard interfaces and protocols. These services are platform-independent. They can be published in public or private directories or repositories for software developers to reuse and to compose their applications. As software architecture, SOA is a conceptual model that concerns the organization and interfacing among the software components (services). It does not concern the development of operational software. The definitions of the Simple Object Access Protocol (SOAP) in 2000, the Web Services Description Language (WSDL) in 2001, and the Universal Description Discovery and Integration (UDDI) in 2002 established the foundation of Web Services (WS) and WS-based architecture, which gave SOA a live instance and paved it road to the success.

The next step of the development is the formation of Service-Oriented Computing (SOC) paradigm. SOC paradigm is based on the SOA conceptual model. However, it goes a step further to include not only the concepts and principles, but also the methods, algorithms, coding, and evaluation considerations, which are a large part of the software development process. As shown in Fig. 1, layers of technologies have been developed for supporting SOC system development, includes

- Service infrastructure for computing, storing, and communication, including Amazon Web Service (AWS) cloud computing infrastructure, IBM Enterprise Service Bus (EBS), developed on the CORBA Object Broker concept, Intel Service-Oriented Infrastructure (SOI), and Microsoft Windows Communication Foundation (WCF).
- Data representation and encoding standards XML, URI, and Unicode data and language definition.
- Object-oriented programming languages C#, Java, and VB, which are used to developing key computing components.
- Service interface and communication protocols, including WEDS, SOAP, and HTTP.
- Ontology framework and languages, including RDF, RDFS, Prolog, and OWL.
- Composition languages, including BPEL, WS-CDL, and the latest Workflow Foundation (WF) and the Visual Programming Language (VPL) from Microsoft; object-oriented languages could also be used as composition languages.
- Service directory and repository standards UDDI and ebXML.

The QoS and dependability features behind the functionality will be discussed in the next section.

The object-oriented computing paradigm is the basis of SOC paradigm. Services are really wrapped objects written in object-oriented programming languages. Although C#, Java, and VB can also be used for application developments, the use of the higher level of composition languages highlights the new SOC paradigm and further distinguishes the application builders from the service developers. Fig. 2 shows the development chart of composition languages. These languages facilitate the composition of more complex services or applications without using a traditional programming language. Particularly, these composition languages are usually supported by visual composition tools, which allow the application builders focus more on the application logic, instead of attending the implementation details of languages and the infrastructure below.

Service-Oriented Development (SOD) concerns the entire software development cycle based on SOA concepts and SOC paradigm, including requirement, modeling, specification, architecture design, composition, service discovery, service implementation, testing, evaluation, deployment, validation, and maintenance. SOD also involves using the current technologies and tools to effectively produce operational software [1]. The immediate predecessor of SOD is the distributed object-oriented software development. The widely used development environments include CORBA (Common Object Request Broker Architecture) defined by OMG (Object Management Group) and Distributed Component Object Model (DCOM) developed by

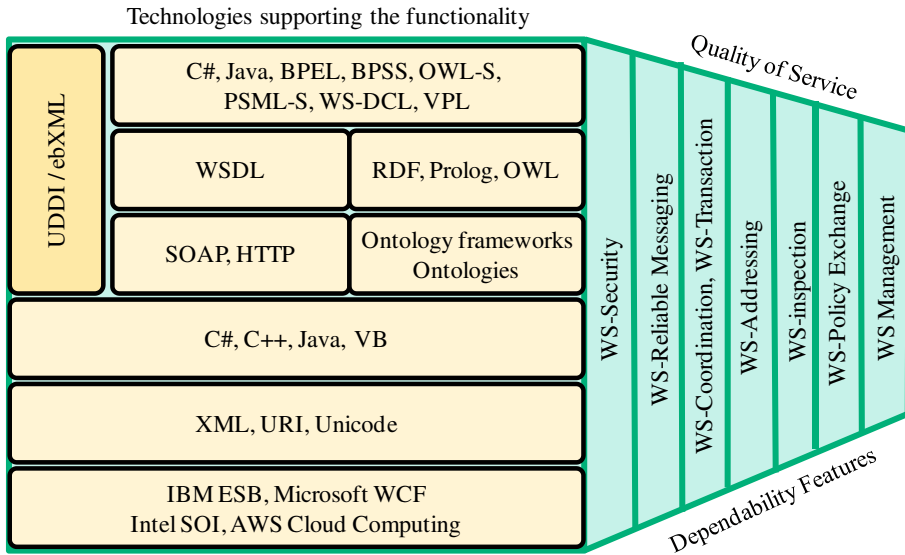


Fig. 1. Technologies and dependability attributes for service-oriented systems.

Microsoft. The successful application of SOD owes to the development effective service-oriented development environments, including IBM's WebSphere, Sun Microsystems' Java Enterprise Edition, Microsoft's Visual Studio, and Oracle's SOA Suite. These development environments provided technologies and tools, including visual composition tools, to support the functional development as well as the full implementation of quality of service and features of system dependability. Fig. 3 shows the visual composition of a service-oriented system using JDeveloper in Oracle's SOA Suite. The visual tool encapsulates constructs, internal activities, and external services into boxes of functional modules. A service-oriented system can be simply composed by dragging and dropping the boxes into the design panel, connecting them, and configuring them with necessary parameter values. The maturity of the development tools largely determines the usability and dependability of the systems developed. The complexity of the software has exceeded intelligence of any individuals, and the tools represent the collective intelligence of human intelligence.

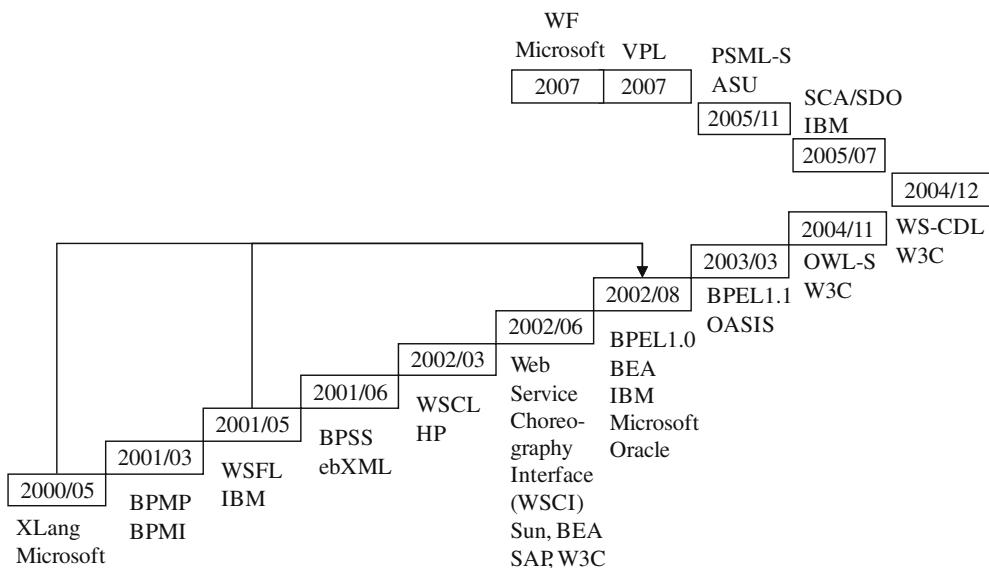


Fig. 2. Development of composition languages for service-oriented systems.

Service-oriented computing and service-oriented software development have been adopted and supported by all major computer companies, including BEA, Google, HP, IBM, Intel, Microsoft, Oracle, SAP, and Sun Microsystems, and their technologies have been standardized by OASIS, W3C, and ISO [2]. Government agents such as US Department of Defense also adopted service-oriented architecture in developing military applications [3].

## 2. Dependability features and quality of service

Dependability of a system is defined to be the system’s ability to deliver specified services to the end users so that they can justifiably rely on and trust the services provided by the system [4]. Dependability includes three aspects of a system: attributes of dependability, means of improving dependability, and impairments that need to be controlled in dependability design. The dependability attributes include safety, vulnerability, confidentiality, data integrity, reliability, availability, and maintainability. The first four attributes are also called security attributes.

Another related concept is the Quality of Service (QoS), which is based on ISO (International Standard Organization) 8402 [1986]. QoS defines quality as “the totality of features and characteristics of a product or service that bears on its ability to meet a stated or implied need.” This is a high-level and generic definition. Different fields have different interpretations of the quality of service. For example:

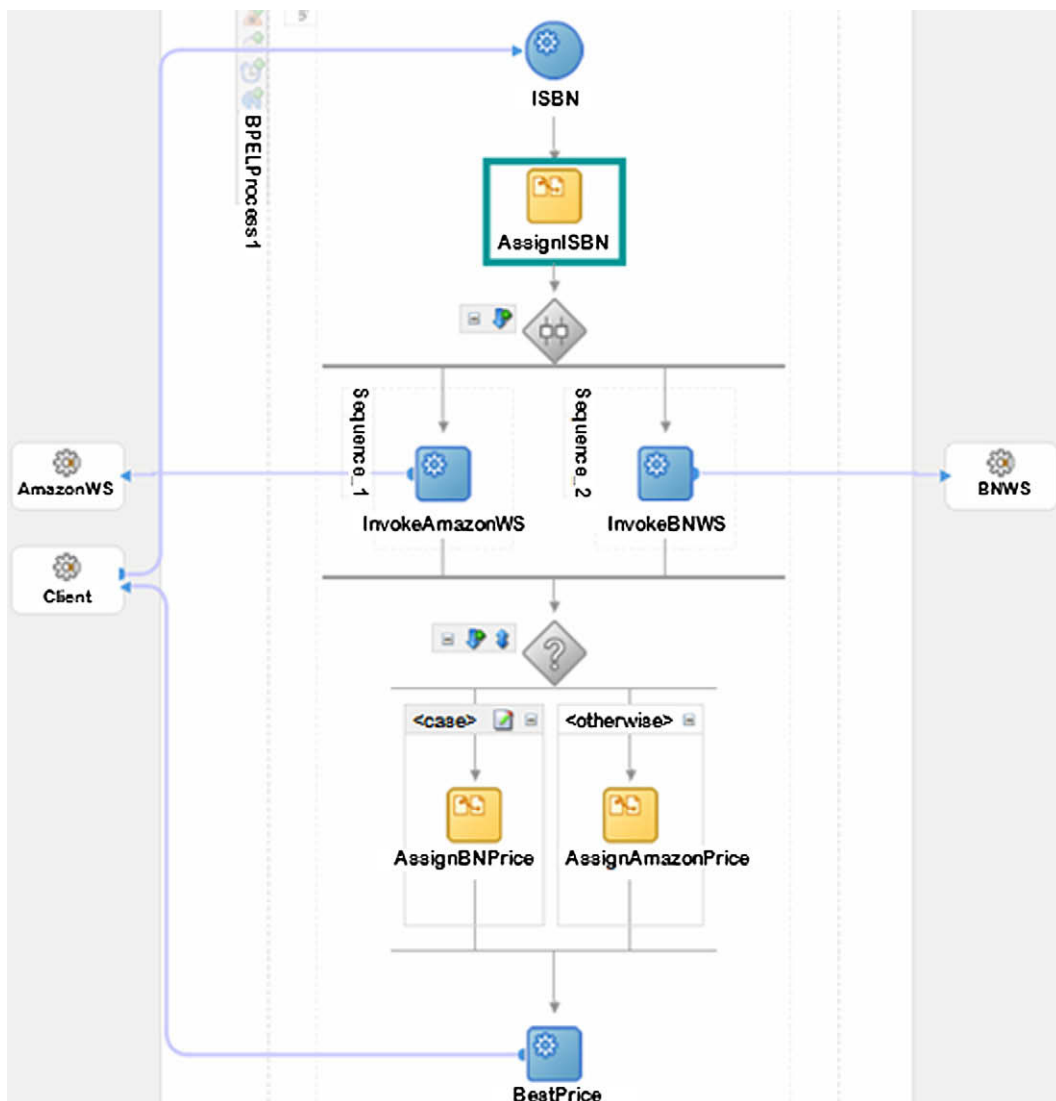


Fig. 3. Visual development using Oracle SOA Suite.

- Network quality: represents the transmission rates, error rates, and other characteristics that can be measured, improved, and, to some extent, guaranteed in advance.
- Software quality: is the degree to which software conforms to quality criteria. Quality criteria include:
  - o Economy, correctness, resilience, integrity, reliability.
  - o Usability, documentation, modifiability, clarity.
  - o Understandability, validity, maintainability, flexibility.
  - o Generality, portability, interoperability, testability.
  - o Efficiency, modularity, reusability.

Although QoS can also include the performance and functionality of a system, it is more often used to characterize the non functional features. In this sense, QoS has the similar meaning as the dependability attributes. However, QoS is more a perspective from the user's point of view, while the dependability attributes are more a perspective from the designer's point of view. The QoS and dependability features for SOC systems are also shown in Fig. 1, which ensure the functionality is delivered to meet the stated or implied need.

All the SOC development environments support the dependability features shown in Fig. 1. For example ASP .Net and WCF security mechanisms include [5]:

Authentication WCF implemented multiple authentication methods, including ASP .Net security, IIS security, integrated Windows security, and Forms-based security. In addition, it supports the X.509 certificate, which employs the public key infrastructure (PKI) to encrypt message to generate digital signature as credential. X.509 specifies standard formats for public key certificates and a certification path validation algorithm.

Authorization WCF supports multiple authorization methods, including access control list and role-based authorization. In addition, it supports XSI standard, which is a claim-based authorization method.

Transport security protects data during their transportation on the network. SSL (Secure Socket Layer) is the common mechanism used for encrypting the data in HTTPS protocol. Transport security depends on the mechanism that the binding the user has selected. For example, if wsHttpBinding is used, the security mechanism is Secure Sockets Layer (SSL).

Message security ensures confidentiality of message by encrypting and signing message before sending them to the transport layer, regardless if the transport layer will encrypt the data to be transported. Web applications often use cookies to store user information on the client machine. The encryption and digital signature can ensure that the cookies are not readable or modifiable outside the application.

SOC applications are distributed and in many cases are Web and Internet-based. Not only security, but also reliability is a major concern, and the models and methods will be different from the traditional ones.

WS-Reliability defined by OASIS [6], is a SOAP-based specification that fulfills reliable messaging requirements critical to SOC applications of Web Services. SOAP over HTTP is not sufficient when an application-level messaging protocol must also guarantee some level of reliability and security. This specification defines reliability in the context of current Web Services standards. WS-reliability specification defines the following reliability features:

- Guaranteed message delivery, or At-Least-Once delivery semantics.
- Guaranteed message duplicate elimination, or At-Most-Once delivery semantics.
- Guaranteed message delivery and duplicate elimination, or Exactly-Once delivery semantics.
- Guaranteed message ordering for delivery within a group of messages.

SOC system testing is another key research area where new models and methods need to be developed. A SOC system consists of services developed by different providers. How can such a system be tested before being deployed? Furthermore, there may exist many services that deliver the same functionality. How can a large number of services be tested and ranked in a group to reduce the testing time?

The papers presented in this special issue will address many of the issues raised in this section in their capacity.

### 3. Coverage of the special issue

This special devotes to designing and evaluating the dependability and QoS features of SOC applications. Five papers that reflect the latest research in assuring these features are selected in this special issue. This section outlines the key contributions of the selected papers. The topics studied in these papers cover the validation of the applications developed in composition languages, QoS ontology and service ranking in terms of QoS features, service monitoring, and the SOC applications in bioinformatics systems and in mission-critical military systems. The summaries of these papers serve as a quick guide to this special issue and are purely based on the guest editors' reading and understanding of the papers. They may not correctly or accurately reflect the authors' work.

#### 3.1. Static validation of WS-CDL documents

There are two major styles of composition languages: orchestration and choreography. BPEL is the main orchestration language widely used. While there are many choreography languages, the Web Service Choreography Description Language

(WS-CDL) is the most significant one recommended by W3C, as shown in Fig. 2. Both BPEL and WS-CDL are XML-based languages.

The paper by Pu et al., presents an approach to statically validate applications developed in WS-CDL. To deal with certain constraints appeared in WS-CDL documents or programs that cannot be captured by their XML Schema definition. The research designed a machine-based constraint language using B method. The language uses abstract machines to represent the constraints of the relations among XML nodes. After modeling the constraints, the corresponding checking algorithms can be designed to implement the static validation of WS-CDL programs. Furthermore, the checking algorithms can be integrated into the WS-CDL editor plug-in for the Eclipse project. The case studies show that the approach is effective and realistic.

### 3.2. A new QoS ontology and its QoS-based ranking algorithm for Web Services

SOC applications are based on an open standards and platform-independent interfaces. It likely to have multiple services available for any given service specification. Thus, it is desired to rank those services so that the application builders can choose the service based on their dependability and QoS features. A number of studies on ranking and selecting available services have been published in the recent years. The paper by Tran et al., proposes a new approach for ranking the services, which is based designing and developing a QoS ontology and a QoS-based ranking algorithm for evaluating Web Services. As we discussed in Section 2, there exist many possible QoS features, and different service providers and application builders may use different QoS features for describing service quality information. This fact leads to the definition of the semantic interoperability proposed in this paper. The proposed QoS ontology can not only describe QoS information in full detail but also facilitate the application builders to express their QoS offers and demands at different levels of expectation. The most promising application of the research is in the automated and dynamic discovery and selection of Web Services in building reconfiguration SOC applications.

### 3.3. Model-based monitoring and policy enforcement of services

As services can be developed by third parties, it is necessary for the application to minor the behavior and the reliability of the services and the overall application. The paper by Bai et al., developed a policy-based runtime monitoring mechanism to continuously watch the quality features. Sensors or probes are instrumented to capture the data and detect anomalies. However, sensors in current software monitoring systems are usually manually instrumented or hard-coded in the program. It is expensive and error prone to implement, and dynamically change the monitors at runtime. The paper proposes a model-based approach for automatic sensors generation and policy enforcement. Web Service interface in WSDL, ontology in OWL, and workflow in OWL-S are used to model the SOC systems. Sensors are generated based on these models from two perspectives: (1) dependency analysis of the data, operations, and services with respect to the ontology model of domain concepts and usage context; and (2) coverage strategies to decide the specific logic and paths to cover and the data to capture by the monitoring sensors. Policies are defined and specified using WS-Policy standards. They are associated to the sensors and are enforced at runtime by the policy engine that interoperates with service execution engine to communicate runtime behavior information and verification results. Experiments show that the automated instrumentation is effective and the system with the sensors that constantly monitor the quality features can perform at acceptable level of performance.

The next two papers will discuss the applications of SOC in business and defense applications, where dependability and quality of services are major concerns.

### 3.4. Business-oriented service modeling

The higher-level of abstraction, flexibility, and productivity in developing SOC systems have benefited many application domains where the experts in the domains can focus on their business logic while getting help from powerful software development tools. In the paper by Han et al., SOC applications in bioinformatics are studied experimentally. They developed a service-oriented problem-solving environment to facilitate the bioinformatics research. The particular approach deployed in their research is called VINCA4Science to business-oriented modeling of service functionalities. The environment makes use of Web Services virtualization and user-centric utilization of modeled artifacts, which can illustrate the practical effects regarding the improvement of domain stability, efficiency, and scalability of bioinformatics systems.

### 3.5. Modeling and simulation of network enabled capability on SOA

Our society is ever more dependent on the correct operations of computer hardware, software, and networks in all aspects. The question is not whether computer systems are dependable enough to be used in mission-critical systems. The question is rather how we can make computer systems more dependable to minimize the impact of impairments. Network Enabled Capability (NEC) is the UK Ministry of Defence's response to the quickly changing conflict environment in which its forces must operate. In NEC, systems need to be integrated in context, to assist in human activity and provide dependable inter-operation. The paper by Liu et al., presents the research on the modeling and simulation of SOA systems and their delivering dependable and sustainable military capability in the systems like NEC. The simulation results indicate that the proposed SOA model can provide a high-level of reliability and sustainability in the provision of capability in a dynamic

environment. The paper also illustrates the use of SOA to achieve the NEC requirements in demonstration system for regional surveillance.

#### 4. Conclusions

Like any computing paradigms, SOC has its strength and weakness, and it develops from infant to maturity. It is our mission to develop concepts, methods, theories, and technologies to help SOC to enter its maturity by applying a more rigorous and more productive development process to make SOC systems more functional and more dependable. The studies presented in this special issue demonstrate the endeavors in this direction, and many more studies and experiments are necessary to make SOC systems as dependable as any human-made systems can.

#### References

- [1] Yinong Chen, W.T. Tsai, Distributed Service-Oriented Software Development, Kendall/Hunt Publishing, 2008. ISBN: 978-0-7575-5273-1.
- [2] Jean-Claude Laprie, On computer system dependability: faults, errors, and failures, in: Proceedings of the 13th IEEE Computer Society International Conference, San Francisco, USA, 1985, pp. 256–259.
- [3] Raymond Paul, DoD towards software services, in: Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, Sedona, 2005, pp. 3–6.
- [4] W.T. Tsai, Miroslaw Malek, Yinong Chen, Farokh Bastani, Perspectives on service-oriented computing and service-oriented system engineering, in: Proceedings of the Second IEEE International Symposium on Service-Oriented System Engineering, 2006, pp. 3–10.
- [5] B. Haidar, Professional ASP .Net 3.5 Security, Membership, and Role Management with C# and VB, Willey, 2009.
- [6] OASIS, Web Services Reliable Messaging TC WS-Reliability 1.1, November 2004. Available from: <[http://docs.oasis-open.org/wsrn/ws-reliability/v1.1/wsrn-ws\\_reliability-1.1-spec-os.pdf](http://docs.oasis-open.org/wsrn/ws-reliability/v1.1/wsrn-ws_reliability-1.1-spec-os.pdf)>.

Yinong Chen  
W.T. Tsai  
*Arizona State University,  
Tempe, Arizona,  
AZ 85298-8809, USA*  
E-mail address: [yinong@asu.edu](mailto:yinong@asu.edu) (Yinong Chen)

Available online 18 June 2009