Service-oriented system engineering: A new paradigm

Conference Paper · November 2005

DOI: 10.1109/SOSE.2005.34 · Source: IEEE Xplore

1 author:

Wei-Tek Tsai
Arizona State University
467 PUBLICATIONS **7,315** CITATIONS
SEE PROFILE

Service-Oriented System Engineering: A New Paradigm

W. T. Tsai
Department of Computer Science and Engineering
Arizona State University, Tempe, AZ 85255, USA
wtsai@asu.edu

Abstract

In the past a few years, we witnessed a rapid progress in Service-Oriented Computing (SOC), which represents a paradigm shift from the current mainstream Object-Oriented Computing (OOC) paradigm to the SOC paradigm. This paradigm shift is changing the way we develop and use software and hardware. Conferences, journals, books, research, experimentation, tools, and products in SOC, Service-Oriented Architecture (SOA), Service-Oriented Enterprise (SOE), Service-Oriented Infrastructure (SOI), Web Services (WS), and associated protocols and standards have emerged and a solid foundation for the new paradigm is being grounded. This paper focuses on the system engineering issues in the new paradigm.

1. Introduction

In the traditional software development paradigm, the developer takes the requirements, converts them into specification, and then translates the specification into the executable that meets the requirements. Several approaches are available to translate the requirements into an operational system including the waterfall model, incremental development, object-oriented computing (OOC), component-based computing. Each approach has its own engineering processes and techniques.

Service-Oriented Computing (SOC) is a new paradigm that evolves from the OOC and component-based computing paradigms by splitting the developers into three independent but collaborative entities: the application builders (also called service requestors), the service brokers (or publishers), and the service developers (or providers). The responsibility of the service developers is to develop software services that are loosely coupled. The service brokers publish or market the available services. The application builders find the available services through service brokers and use the services to develop new applications. The application development is done via discovery and composition rather than traditional design and coding.

In other words, the application development is a *collaborative* effort from three parties: application builders, service developers, and service brokers [9]. Furthermore, services are platform-independent and loosely coupled so that services developed by different providers can be used in a composite service. Many standards have been developed to ensure the interoperability among services. On the other hand, the competition is fierce. Only the best services can survive because many service providers, for a given known service requirement, for examples, password encryption and "add-to-cart" services, many providers can implement and publish the same service for application builders to use in their applications.

SOC is being adopted by different applications, e.g., electronic business has been based on SOC from the beginning. Large enterprises and mission-critical applications such as Command-and-Control (C2) systems are also moving into the SOC paradigm [5].

2. Service-Oriented Enterprise

An enterprise system often has applications and a stack of infrastructure including databases, operating systems, and networks. SOC is applicable to all of these layers. This can be illustrated by the Service-Oriented Enterprise (SOE) being standardized by OASIS [2][4], as shown in Figure 1. It shows a layered system of an SOE with SOC architecture at every layer. Typically, the application is implemented as Web services. The top layer of the system architecture is the SOE. The next layer is the SOA or ebSOA (SOA for electronic business) which is a version of SOA adapted to electronic business. The next layer is the Service-Oriented Management (SOM), which implements the non-functional features such as fault-tolerant computing, reliability, security, and policies. Service-Oriented Infrastructure (SOI) provides virtual services that represent the services that can be provided by hardware components. For example, Intel is developing this layer to map its hardware layer resources, including computing resources, memory resources, networking resources, devices, sensors, and actuators, to the service-oriented architectures above. In [9], a service-oriented operating system is proposed.

In summary, the entire stack of computing and communication infrastructure can be structured in an SOC manner.

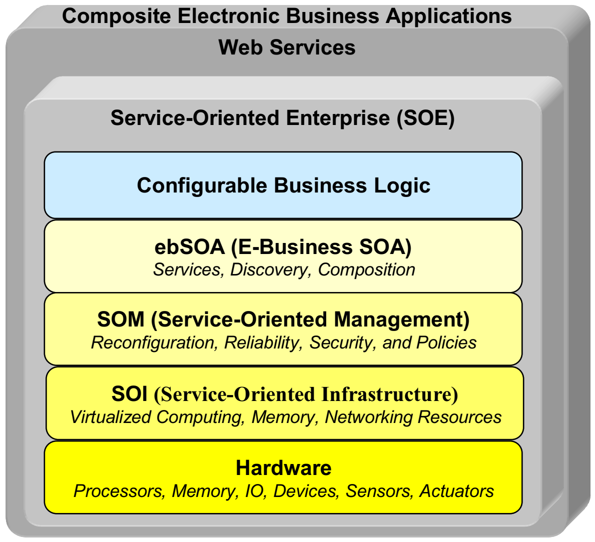
Composite Electronic Business Applications
Web Services
Service-Oriented Enterprise (SOE)
Configurable Business Logic
ebSOA (E-Business SOA)
Services, Discovery, Composition
SOM (Service-Oriented Management)
Reconfiguration, Reliability, Security, and Policies
SOI (Service-Oriented Infrastructure)
Virtualized Computing, Memory, Networking Resources
Hardware
Processors, Memory, IO, Devices, Sensors, Actuators

Figure 1. SOE framework

3. Service-Oriented System Engineering

Current research and practice on SOC is largely focused on functionality and protocols of SOC software because SOC development starts from the application software layer. However, as SOC moving into mission-critical applications, as well as the entire computing and communication infrastructure moving to SOC, Service-Oriented System Engineering (SOSE) issues need to be addressed. SOSE emphasizes on the specification, analysis, fault-tolerant computing, verification and validation, model checking, testing, dependability evaluation of service-oriented software and hardware.

Table 1 lists typical SOSE techniques in each of development phases. Many of the techniques are collaborative. For example, test cases may be contributed in a collaborative manner by all three parties: the service provider can provide sample unit test cases for the service broker and service requestors to reuse; the service broker can provide its own test cases via specification-based test case generation tool, and the broker may even make the tool available for all the parties; the application builder can examine the sample test cases by the service broker, apply the test case generation tool provided by the service broker, and even contribute its own application test cases.

Table 1. Deferent SOSE techniques

Development phase	SOSE techniques
Collaborative specification & modeling	Service specification languages, Model-driving architecture, Ontology engineering, and Policy specification.
Collaborative verification	Dynamic completeness and consistency checking, dynamic model checking, and Dynamic simulation.
Collaborative design	Ontology engineering, dynamic reconfiguration, dynamic composition and re-composition, dynamic dependability (reliability, security, vulnerability, safety) design
Collaborative implementation	Automatic system composition and code generation
Collaborative validation	Dynamic specification-based test generation, group testing, remote testing, monitoring, and dynamic policy enforcement
Collaborative run-time evaluation	Dynamic data collection and profiling, data mining, reasoning, dependability (reliability, security, vulnerability, etc) evaluation
Collaborative operation and maintenance	Dynamic reconfiguration and re-composition, dynamic re-verification and re-validation

While the basic engineering principles remain the same, the way they are applied will be different in the SOC paradigm. Specifically, most engineering tasks will be done on the fly at runtime in a collaborative manner. Because systems will be composed at runtime using existing services, many engineering tasks need to be performed without complete information and with significant information available just in time before application. In this way, SOSE in some way may be drastically different from traditional engineering where engineers have complete information about the system requirements and thorough analyses can be performed even before system design is started.

4. Service-Oriented Reliability Evaluation

This section uses an SOSE example, i.e., reliability evaluation of service-oriented applications, to illustrate the major differences between traditional system engineering and SOSE.

Current reliability evaluation follows model development, model validation, and model application. The primary task is to identify or develop a model that will fit the application data collected (profile). This approach has been used since 1970's. For component-based software, component-based reliability models are proposed to compute the system reliability based on the reliabilities of the components and the structure of the system [1][3]. The reliability evaluation for SOC software is different, and has the following characteristics:

- Dynamic model construction after a new service is composed or a re-composition that changes the structure of an existing composite service. The objective is not to reconstruct the entire model so that the failure history can be reused in the restructured model.
- Determine the optimal window size of history, i.e., how far the historic failure data should be used.
- Dynamic reliability computation based on the incremental update of failure data, reliability, structure.

Figure 2 outlines the new Data-driven Reliability Evaluation Process (DREP) and its evaluation process. This is a new approach of reliability evaluation because its focus is no long on the reliability model. Instead, it focuses on the entire process from runtime data collection to reliability evaluation:

- The atomic (basic) services have been tested by their service providers. A set of test cases, their oracles, and reliability (TOR) are provided by the service providers.
- When a service is registered to a service broker, the broker will evaluate the service before registering it. If many services that meet the same specification are being registered, group testing technique can be applied [8]. More test cases can be generated by specification-based test generation technique. Oracles can be established and new reliability data can be computed. The services with acceptable reliability will be stored or listed in the service repository of the broker. Service ranking can also be provided.
- When composing a new service using the services in the repository, automated code generation can be applied. Code instrumentation can be done to allow failure data collection. The data can be used to update the reliability of the constituent services.
- Based on composition structure of the composite service and the individual reliabilities of the constituent services, the reliability of the composite service can be computed using a Service-Oriented Reliability Model (SORM) [7].
- If the composite service is reconfigured or recomposed, the SORM model can be restructured automatically and the reliability of the composite service can be re-computed.

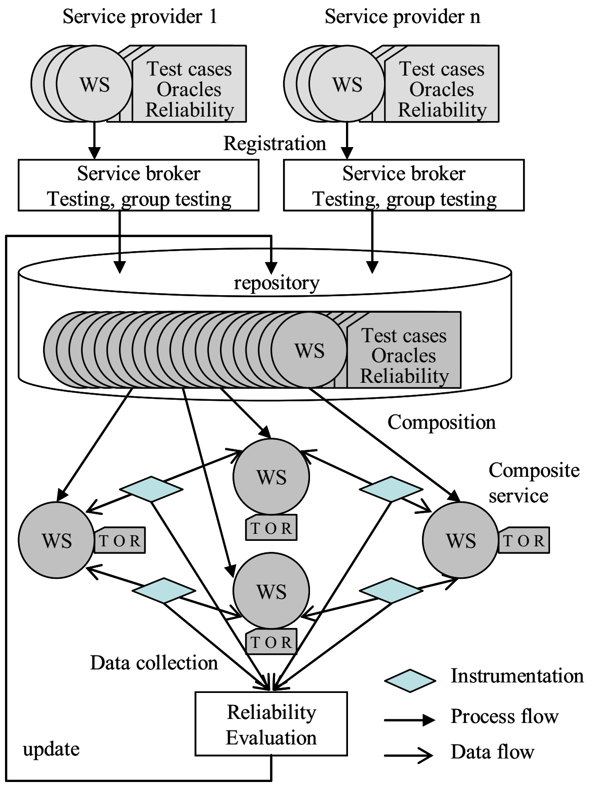

Figure 2. Dynamic reliability evaluation process

The key difference between DREP and traditional reliability modeling is that it emphasizes on data collection and dynamic profiling rather than using historical data. The reliability is continuously updated as the SOC software is re-composed to meet the changing environment. As an SOC application may consists of services from different providers, the reliability is computed in a collaborative manner using current data just collected and profiled from participants. Thus, the DREP uses a just-in-time collaborative approach. Table 2 summaries the differences.

Table 2. Traditional models vs. DREP model

Element	Traditional software reliability evaluation	DREP for SOC software
Data and source	Data are supplied by major computer corporations or government agencies.	Data are dynamically collected from monitoring services at runtime, or obtained by dynamic evaluation performed at runtime.
Profiling	Profiling using data from various sources.	Dynamic profiling based on runtime data collected.
Reliability models	Key component of reliability engineering, and use data to validate the model and to estimate model parameters.	Reliability models are important but the emphasis is on using current data and profiles to accurately estimate reliability.
System recon-figuration and re-composition	If the system is changed, it is essentially considered a new system and a new model needs to be developed.	The reliability model can be dynamically restructured and data collected before restructuring can partially be reused.

5. SOSE is Different

SOC is a new paradigm for computing and thus new engineering techniques needed to be developed to make SOC software and hardware dependable, reliable, safe, and secure. SOSE techniques are different from traditional system engineering techniques even though the basic engineering principles such as mathematics remain the same. Due to the dynamic features such as runtime composition and re-composition, new applications may not be evaluated by traditional system engineering because many components may be dynamically discovered and composed, and their source code may not be available. Thus, dynamic runtime system engineering techniques need to be applied. This paper uses a reliability engineering approach to highlight the differences.

Even though this paper uses mainly software to illustrate SOSE, the same can be applied to hardware and networks. Major computer companies are developing SOI and SON (Service-Oriented Networks) to support SOC applications at this time. They will need to develop the related SOSE techniques.

References

[1] D. Hamlet, D. Mason, D. Woit, “Theory of Software Reliability Based on Components”, 23rd International Conference on Software Engineering (ICSE'01), 2001, pp. 361-370.

[2] Intel: Service-Oriented Enterprise, The Technology Path to Business Transformation, http://www.intel.com/business/bss/technologies/soe/soe_backgrounder.pdfpresentations/

[3] B. Littlewood, L. Strigini, “Software reliability and dependability: a roadmap”, Proceedings of the Conference on the Future of Software Engineering, 2000, pp. 175 - 188.

[4] OASIS ebSOA proposal: http://www.oasis-open.org/committees/ebsoa

[5] R. Paul, “DoD Towards Software Services”, Tenth IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS 05), February 2005, pp. 3-6.

[6] M. P. Singh, M. N. Huhns, *Service-Oriented Computing*, John Wiley & Sons, 2005.

[7] W. T. Tsai, D. Zhang, Y. Chen, H. Huang, R. Paul, N. Liao, “A Software Reliability Model for Web Services”, the 8th IASTED International Conference on Software Engineering and Applications, Cambridge, MA, November 2004, 144-149.

[8] W. T. Tsai, Y. Chen, Ray Paul, Hai Huang, Xinyu Zhou, Xiao Wei, "Adaptive Testing, Oracle Generation, and Test Script Ranking for Web Services," in Proc. 29th Annual International Computer Software and Applications Conference (COMPSAC), Edinburgh, July 2005, pp 101-106.

[9] W.T. Tsai, Y. Chen, R. Paul, "Specification-Based Verification and Validation of Web Services and Service-Oriented Operating Systems", Tenth IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS 05), Sedona, February 2005, pp.139 - 147.