

Consumer-Centric Service-Oriented Architecture: A New Approach

W.T. Tsai, Bingnan Xiao, Raymond A. Paul*, Yinong Chen
Arizona State University, Tempe, AZ 85287-8809, USA
*Department of Defense, Washington, USA
wtsai@asu.edu

Abstract

This paper extends the current Service-Oriented Architecture (SOA) to introduce a Consumer-Centric Service-Oriented Architecture (CCSOA) paradigm. The current SOA is producer-centric, because the basic idea is that service providers publish services that they produce and let the consumers to search available services to compose their applications. CCSOA focuses on consumers' publishing the services they need and even the applications they need. The service providers must produce services that are in need. This new paradigm extends the design and code sharing, and thus further improves the software productivity. This paper presents the concepts, architecture, enabling techniques, and illustrative examples.

1. Introduction

Service-Oriented Computing (SOC) including service-oriented architecture (SOA) and Web Services (WS) emerges as a new computing paradigm. Major computing corporations and government agencies are adopting this technology. The idea that a program as a service that can be discovered, matched, composed, executed, verified, and monitored at runtime forms the new paradigm. In SOA, service providers register the services in a service broker and the service consumers (application builders) search and discover required services by the broker. Once a required service is found, the service consumer can bind the service into its application. Service collaboration is essential in SOA because a complex service can compose of several other services.

Consumer-Centric SOA (CCSOA) is a new architecture designed to support collaboration-oriented service composition. In addition to publishing service specifications in SOA, CCSOA also publishes application collaboration specifications for discovery,

matching, and subscription. The CCSOA is different from the conventional SOA framework as presented in [3][12]. In conventional SOA, service providers develop and publish their services, and consumers are responsible to discover the right service published as well as use the service in their applications. This is really a *producer-centric* approach of SOA. Even though SOA needs all three parties: service providers, consumers, and brokers, the focus is on the producers. Specifically, the majority of support is for producers, e.g., the broker will list registered services, the specification techniques and standards are designed so that producers may produce the right software for a given specification. There is no corresponding broker for consumers, nor is the corresponding application specification published.

In CCSOA, consumers publish their application requirements together with associated service specifications including the workflow. Once these are published as application templates, any service providers can submit their software or services to meet the application requirements. This way of computing is consumer-centric because now the service providers will look for application needs.

Furthermore, a CCSOA platform can be developed on top of an SOA platform, and both can be implemented and run concurrently without conflict with each other. When a consumer publishes an application, the followings will be published to ensure successful application development:

- The entire application workflow;
- Service description of participating services;
- Acceptance criteria for each service;
- Acceptance criteria for integration and collaboration within the application; and
- Acceptance criteria may include both functional and nonfunctional acceptance criteria.

2. CCSOA Overview

Specification techniques have been used to specify services from the beginning of SOA. We can divide the specification techniques into four stages:

1) Initial stage or input/output stage

Service descriptions mainly contain the input/output information, such as function name, return type, and parameter types. This is the starting point for service specification, but information contained is not sufficient for a variety of detailed analyses [13]. It is possible to perform dynamic composition in this stage, but the designer needs to be concerned with lots of issues to ensure successful application development. This is a purely *producer-centric* approach.

2) Process description stage

Service specifications contain an abstract process description in addition to input/output information. A variety of process description languages are available such as BPEL [7], OWL-S [8], and PSML-S [9]. With internal process description, it is now possible to do detailed analyses that were not possible before. Tasks such as completeness and consistency (C&C) analysis, model checking, composition reasoning, workflow analysis, abstract path generation, abstract execution tracing, and test case generation [13][15] are now possible. Using service specifications in this stage, dynamic composition becomes easier and many related tasks can be automated. It is also possible to perform various dynamic V&V once the SOA application is specified. However, it is still the responsibility of the application builder to design the overall workflow of the application. In many ways, this is still a *producer-centric* approach, because the focus is on the features and functionality of services, not on how they are used by an external agent or user.

3) Service collaboration specification stage

The service specifications not only contain everything in the previous two stages, but also service collaboration protocols such as collaboration protocol establishment, collaboration protocol profiles, collaboration constraints [4][5], and patterns [14]. With these descriptions, tasks such as dynamic collaboration establishment, dynamic and automated application composition, and dynamic collaboration verification can be performed. Some well-known protocols include CPP/CPA (Collaboration Protocol Profile/Collaboration Protocol Agreement) [6] and *use scenarios* [14]. Dynamic composition can now be established by two or more services after they interact with each other to establish a specific collaboration at runtime. This must be done by the application builder

before, but now can be performed at runtime by software. This is a transition technology from producer-centric specifications to consumer-centric specifications because specifications now describe how others can use the service in addition to what it does.

4) Consumer-centric specification stage

In this stage, the entire application can be specified, published, searched, discovered, and composed. This is consumer-centric because it focuses on consumers and their applications. In addition to searching services, a consumer can now search to see if the needed application may be available already, and if not, the consumer can publish the application specification and ask application developers to bid for a contract.

2.1 CCSOA Framework

CCSOA provides a framework for collaboration-oriented service specification, registration, discovery, matching, verification, validation, and composition. The CCSOA architecture is shown in Figure 1.

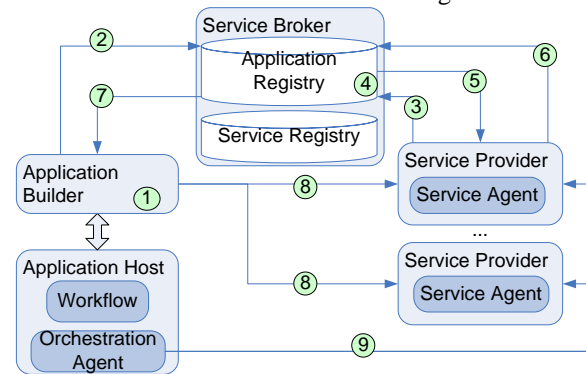


Figure 1: CCSOA Architecture

A CCSOA broker stores not only service specifications, but also application templates and collaboration patterns. Service providers can subscribe to this application registry. Once a consumer publishes an application template to the registry, the subscribed service providers will be informed and they can develop services and application for the new application. Once a new service is developed for a published template, the consumer will be informed to test and evaluate the newly available service.

The numbers in the diagram shows a scenario how the three parties collaborate to build an application:

1. An application builder (consumer) develops an application template using BPEL, SCA/SDO [19] [20], or PSML-S, which includes information on workflow specification, service specification, service acceptance criteria, and application acceptance criteria.
2. The application template is registered and

published to a service broker and published in the application registry.

3. A service provider subscribed to the application registry is informed the availability of new templates and queries the new application template.
4. The ontology and standard taxonomy can help automatic matching between the requested and registered application templates.
5. Once the service broker finds an application template in its registry, it returns the application template's details to the service provider.
6. The service provider develops a service according to the application template and submits it to an application template. Each service submitted to the broker will be evaluated by the service acceptance criteria.
7. Once a service passes the evaluation, the service broker will notify the application builder that a service is available for the published template.
8. Using the binding information from the service broker, the application builder test and evaluate the service.
9. If the services pass the application acceptance testing, the application builder will bind the service into the target application.

Once the application is completed, new service can still arrive. For example, a new service with lower cost or higher performance may be available after the application is completed. The consumer can still take advantage of these new services by going through the service evaluation and application evaluation again from time to time. In this way, the cost of owning the application keeps on going down and performance going up as new services become available.

The service code can be imported to the consumer's site, or reside on the service broker's site, or on the service provider's site. The application hosting site needs to have an orchestration engine to control the workflow of the execution, as shown in Figure 1.

There are two levels of acceptance testing: service acceptance and application acceptance. Once the participating services are all evaluated using the service acceptance criteria specified by the service broker, these services will be eligible for being sent to the application builder. Then they are evaluated using the application acceptance criteria specified by the consumer. Once all services needed are available, the application will be composed and sent to the application host. The application host gets the workflow information from application builder and orchestrates the collaboration of the participating services through the orchestration agent.

2.2 Publishing Application Templates

When building a new application, consumers can publish the application template including workflow specification and service description. The application template may have not been finalized yet at that time. Minor variations are always encouraged to be made to the template so that the application template can be refined by peers to get an optimized specification.

Even if a application template is finalized, multiple implementations satisfying the template may be available and the consumer needs to make an intelligent decision on the final selection, and the selection can be by dependability, performance and user feedbacks.

Once a complete application is available, it is possible to evaluate the application even if the consumer does not have the access to the source code. However, it is necessary for a consumer to publish reasonable and qualified application templates, otherwise it may be impossible for service providers to understand and fulfill the requirements. Before the publishing the application templates, the consumer can develop the application model using a process modeling language such as BPEL4WS or PSML-S [9]. PSML-S provides a specification and modeling language that supports multiple analyses based on a single integrated model, i.e., SMMA (Single Model with Multiple Analyses) [9] and a simulation framework DDSOS (Distributed Dynamic Service-Oriented Simulation) [17]. Once the model is specified for the application, simulation can be performed using DDSOS to evaluate the soundness of the application design. Only if the simulation results are satisfactory, the consumer will publish the application template.

2.3 Handling Application Evolution

Software keeps on evolving, and this may cause issues in CCSOA. Several alternative mechanisms are available to handle this problem:

- Make each variant as a new application template.
- Publishing parameterized service description.
- Use a composite factory service.
- Use Inheritance.

3. Key Techniques in CCSOA

3.1 Collaboration Descriptions

Collaboration descriptions provide information on the collaboration capabilities of a service and how

collaboration can be established at runtime between services that do not know each other before their interaction. One such technique is the use scenario. It is a specification of how the service can be used by other services in PSML-S. It does not specify the service's internal control logic but focuses on how the service can be used by other services. To be more specific, a use scenario indicates the temporal logic and possibly also timing constraints on a sequence of function calls. It provides a usage pattern on how to invoke the interfaces of the service. Collaboration may have variations. Similar to the inheritance in object-oriented programming, collaboration can also have inheritance.

3.2 Discovery and Matching

In addition to the service discovery and matching in traditional SOA, CCSOA also has discovery and matching for application templates and/or collaboration patterns. CCSOA discovery and matching is based on *collaboration ontology*, i.e., a repository of various collaboration patterns and classifications. A collaboration ontology is similar to a service ontology except the information is about service collaboration rather than service. Service providers can find appropriate templates and/or collaboration patterns that they support in a collaboration ontology.

3.3 Verification and Validation in CCSOA

Before a service provider can register a service to support an application template/collaboration pattern published by an application builder, the service need to be verified and validated by the service broker against the service acceptance criteria.

Each service broke has a Service Verification and Validation Agent (SVVA) that can verify and validate services. When one service provider wants to subscribe to a application template/collaboration pattern, the SVVA firstly retrieves the test cases provided by the application builder or other parties from the repository. Then it needs to perform unit testing on the service being registered. In addition to the general unit testing for functional validation, test agent may need to perform other property-specific testing according to different requirements from the application builder. The property-specific testing could include reliability testing, security testing, robust testing, performance testing. Once the unit testing and property-specific testing are done, test agent needs to perform the interoperability testing to validate the collaboration capabilities of this service. Only if the subscribing service passes all the testing, it can register to the

application template/collaboration pattern. Whenever changes are made to the services, regression testing needs to be performed to validate the revised component services. When composing a new service, Collaborative V&V [15] technology can be applied for integration testing in a distributed environment.

3.4 Application Classification

In CCSOA, entire applications together with associated services will be represented in application templates, and these application templates can be stored in an ontology manner so that both consumers and producers of services can search, navigate and discover applications and their associated services. First, the applications in an ontology database can be categorized into a classification tree so that relationship between these applications can be visualized to facilitate easy access to these applications. Second, applications can be grouped into various Communities of Interest (COI) according to various trades or professions. For example, a nanotechnology COI may contains various applications associated with nanotechnology so that both consumers and producers of nanotechnology can have easy access. Third, the applications can be ranked according to various criteria such as:

- Ranked by the number of usages.
- Ranked by quality metrics such as reliability, security, fault-tolerance and performance.
- Similar ranking can be applied to rank individual services in addition to applications in CCSOA.

3.5 Linking Applications and Services

In CCSOA, when a service implementation is accepted to be an acceptable member of an application, the CCSOA infrastructure can maintain the link between the application and service implementations. Because multiple service implementations can serve the same service specification, the CCSOA infrastructure can maintain this *one-to-many* links from the application to multiple service implementations. Similar, a service implementation may be able to contribute to multiple applications, and the CCSOA infrastructure can also maintain this *one-to-many* links, this time from a service specification or service implementation to applications. Like SOA, the CCSOA can also provide *one-to-many* links from a service specification to multiple service implementations.

If two service specifications often appeared in same applications, one can deduce that these two services are indeed related to each other. This information is useful

for both consumers and producers. A consumer may use this information when designing a new application. Once a given service is chosen, its closely related services may need to be examined for possible inclusion in the new application. A producer may use this information too, in verifying a given service implementation, the producer may perform selective integration testing using the service and its closely related services together.

3.6 Rapid Application Generation in CCSOA

With CCSOA, if sufficient number of applications and their associated services are available, together with associated SOSE (Service-Oriented System Engineering) [16] tools, it is possible for an application designer to develop a large application with millions of code rapidly. The application process can be outlined as follows:

1. The designer looks into the application ontology database to look for an existing application template that fits the application needs;
2. If an existing application fits the needs, the designer can simply use the application;
3. If an application largely fits but some customization is needed, the designer needs to design several new service specifications and customize the application template for the new applications;
4. The new application will be evaluated using SOSE techniques including simulation, model checking, C&C checking, and testing using the newly created service specification and other existing service implementations that are associated with the application;
5. As only few services in the new application template will need to be developed, the designer can go ahead and implement these new services, have a contractor to develop the services, or look into the service repository to see if they have been developed before.
6. Once the service implementations for the new services are available, the new application can be evaluated using SOSE rapidly before deployment.

As shown in Figure 2, each collaboration party has multiple and potentially unlimited choices of services that are registered as the service providers for the same type of process. The services can be dynamically verified and validated for interoperability and integrity before they register themselves as the candidate services. Once the services are registered, the application can be composed rapidly.

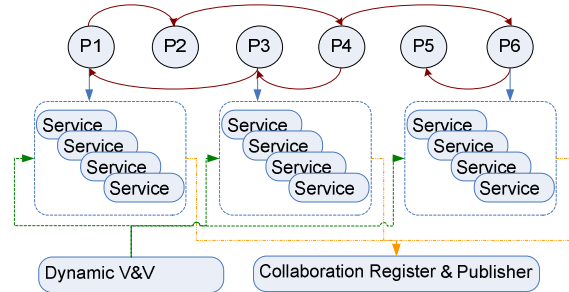


Figure 2: Collaboration pattern and composition

4. CCSOA Example

This section uses an example of a college admission system to illustrate the CCSOA concept. The right part of the Figure 3 shows application process specification of the system. When a user logs in, a security service will be used. Once a user passes the security check, she or he can submit the application package including the scores from language test and other subject tests. The system can verify the language test scores through the services of the test institutions and the subject test scores through the services of schools or exam boards.

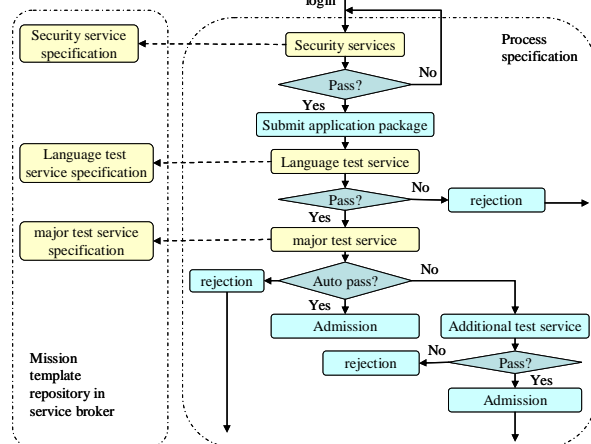


Figure 3: Example of an application specification

Each box in the process specification can be an internal component or an external service. The application builder can publish the specification of the entire application and the specifications of individual service components (left part of the diagram). Multiple service builders can submit services that can meet the application's service acceptance criteria, as shown in Figure 4. The service broker will test the services based on the service acceptance criteria. If a large number of services are produced for the same service specification, group testing technique can be applied [15]. The best services will be recommended to the application builder, which will test the services using application

acceptance criteria. The best services that pass the criteria will be integrated into the application.

5. Summary

This paper presents, compares, and contrasts the concept, architecture, and techniques of producer-centric and consumer-centric SOA and then uses an example to illustrate the new CCSOA. This paper proposes that publishing application and collaboration specifications can significantly improve the software productivity. It presented concrete techniques that support the new paradigm, including collaboration description, discovery and matching techniques, verification and validation of the specifications and implementations, and rapid and dynamic application generation through automated code generation.

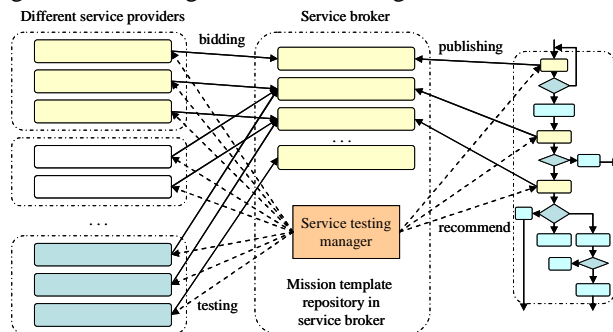


Figure 4: Services following application template

6. References

- [1] X. Fu, T. Bultan, and J. Su, "Formal Verification of E-Services and Workflows," Proc. Workshop on Web Services, E-Business, and the Semantic Web (WES), LNCS 2512, Springer-Verlag, 2002, pp. 188–202.
- [2] H. Huang, W. T. Tsai, R. Paul and Y. Chen, "Automated Model Checking and Testing for Composite Web Services", 8th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC), Seattle, May 2005, 300-307.
- [3] Intel: Service-Oriented Enterprise, The Technology Path to Business Transformation. http://www.intel.com/business/bss/technologies/soe/soe_bac_kgrounder.pdf
- [4] Nickolas Kavantzias, David Burdett, Tony Fletcher, Yves Lafon, "Web Services Choreography Description Language (WS-CDL)", Version 1.0, W3C Working Draft 17 December 2004. <http://www.w3.org/TR/ws-cdl-10/>.
- [5] Frank Leymann, Web Services Flow Language, Version 1.0, May 2001. <http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>
- [6] OASIS: ebSOA proposal: <http://www.oasis-open.org/committees/ebsoa>
- [7] OASIS: Business Process Execution Language for Web Services (BPEL4WS), 2003. <http://xml.coverpages.org/bpel4ws.html>
- [8] DARPA: OWL-S: <http://www.daml.org/services/owl-s/>
- [9] W.T. Tsai, Ray A. Paul, Bingnan Xiao, Zhibin Cao, Yinong Chen, "PSML-S: A Process Specification and Modeling Language for Service Oriented Computing", The 9th IASTED International Conference on Software Engineering and Applications (SEA), Phoenix, November 2005, pp. 160-167.
- [10] Object Management Group, Unified Modeling Language (UML), version 2.0. <http://www.omg.org/technology/documents/formal/uml.htm>
- [11] Object Management Group, Meta Object Facility (MOF) 2.0 Core Specification. <http://www.omg.org/docs/ptc/04-10-15.pdf>
- [12] M. P. Singh, M. N. Huhns, *Service-Oriented Computing*, John Wiley & Sons, 2005.
- [13] W. T. Tsai, R. Paul, Y. Wang, C. Fan and D. Wang, "Extending WSDL to Facilitate Web Services Testing", Proc. of IEEE HASE 2002.
- [14] W. T. Tsai, R. Paul, H. Huang, B. Xiao, and Y. Chen, "Semantic Interoperability and Its Verification and Validation in C2 Systems", 10th International Command and Control Research and Technology Symposium (ICCRTS), 2005, and the paper is available at www.dodccrp.org/events/2005/10th/CD/papers/200.pdf.
- [15] 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.
- [16] Wei-Tek Tsai, "Service-Oriented System Engineering: A New Paradigm," IEEE International Workshop on Service-Oriented System Engineering (SOSE), Beijing October 2005, pp. 3 - 8.
- [17] W. T. Tsai, Chun Fan, Yinong Chen, "DDSOS: A Dynamic Distributed Service-Oriented Simulation Framework", 39th Annual Simulation Symposium, Huntsville, AL, April, 2006.
- [18] O. Zimmermann, Sven Milinski, M. Craes, F. Oellermann, "Second Generation Web Services-Oriented Architecture in Production in the Finance Industry", OOPSLA'04, Oct. Vancouver, 2004.
- [19] IBM Developer Works, Service Component Architecture, <http://www-128.ibm.com/developerworks/library/specification/ws-sca/>
- [20] IBM Developer works, Service Data Objects, <http://www-128.ibm.com/developerworks/webservices/library/specification/ws-sdo/>