Toward a Quality-driven Service Component Architecture
Techniques and Models

Maryem Rhanoui[1] and Bouchra El Asri[2]
[1]*School of Information Science, Rabat, Morocco*
[2]*National Higher School of Computer Science and Systems Analysis, Rabat, Morocco*

Keywords: Service Component, Service Component Architecture, Critical System, Fault Tolerance, Contracts.

Abstract: Service Component Architecture (SCA) is a recent approach and an industry standard for developing complex and distributed systems. Despite the growing research work it still lacks a formal basis for handling quality and reliability of safety-critical systems. In this paper we present main techniques and models for assuring quality and trustworthiness of component-based systems in general, then we present and justify the choice of the design-by-contract approach that we adopted for the following of our research about SCA-based systems.

1 INTRODUCTION

Service Oriented Architecture is a promising paradigm for developing complex systems that utilizes services as fundamental elements for developing applications, in this perspective; Service Component Architecture (SCA) is a new concept that offers a component model for building SOA architecture.

Official SCA specification document includes SCA assembly model specification (Beisiegel, 2007) and SCA policy framework (Barber). However, as an expanding approach, it still needs more formal models and frameworks for modelling and verifying systems.

Our litterature review shows that most research efforts have focused on technical aspects of SCA, leaving aside the treatment of quality problems and extra-functional properties of service component.

Our field of research focus on the design and development of complex and safety-critical systems. Critical systems (Isaksen et al., 1997) are systems whose failure could cause loss of human lives, cause property damage, or damage to the environment, such as aviation, nuclear, medical applications, etc.

In this paper we present main techniques and models for handling quality and trustworthiness of component-based systems. Among the presented approaches, the contract-based approach is a light-weight formal method for designing quality-driven systems by spefiying its non-functional and quality properties

The remainder of this paper is organized as follows: section 2 will be dedicated to the presentation of the concept of service component.

Section 3 will focus on main techniques and models for assuring trustworthiness of component-based systems.

Finally section 4 will present and justify with our proposed approach.

2 SERVICE COMPONENT ARCHITECTURE

Service-oriented computing (SOC) is the computing paradigm that utilizes services as fundamental elements for developing applications (Papazoglou and Georgakopoulos, 2003). Service Component Architecture (SCA) proposes a programming model for building applications based on components following the SOA paradigm.

SCA offers many advantages: it simplifies development of business component and assembly and deployment of business solutions built as networks of services; increases agility and flexibility, protects business logic assets by shielding from low-level technology change and improves testability.

2.1 Component Model

Various component models of Service Component

DOI: 10.5220/0004094701920196
In *Proceedings of the 14th International Conference on Enterprise Information Systems* (ICEIS-2012), pages 192-196
ISBN: 978-989-8565-11-2

Architecture were proposed in literature.

For Ding (Ding et al., 2008) proposed component model, a service component provide and require services. A service can be described by operation activities as by well-defined business function. A *component* provides and consumes services via *ports*.

A **port** *p* is a tuple (*M,t, c*), where *M* is a finite set of methods, *t* is the port type and *c* is the communication type.

A **component** *Com* is a tuple (*Pp, Pr,G,W*), in which *Pp* is a finite set of provided ports, *Pr* is a finite set of required ports, *G* is a finite sub component set

Moreover, Du et al (Du el al., 2008) included contract concept in the Service Component meta-model

A **contract** *Ctr* is a quadruple (*P, Init, Spec, Prot*) where

- *P* is a port;
- *Spec* maps each operation *m* of *P* to its specification (*am,, gm , pm*) where:
  - *am* contains the resource names of the port *P* and the input and output parameters of *m*.
  - *gm* is the firing condition of operation m, specifying the environments under which *m* can be activated.
  - *pm* is a reactive design, describing the behaviour of *m*.
- *Init* identifies the initial states.
- *Prot* is a set of operations or service calling events.

2.2 Framework and Meta-Model for SCA

Zhang et al (Zhang et al., 2009) realized a framework for modelling service-oriented systems. In this framework, the concept of "service component" is considered as a unit of first-class modeling to capture the functional unit linked to the abstract service.

A "service component" is autonomous, self-descriptive and can offer and deliver functionality to other "component service" through interfaces without displaying the implementation details.

The meta-model of the component service (Zhang et al., 2009) is defined from the following aspects: **Identification:** Identifies a service component, **Specification:** Declares the service interfaces and service contracts, **Content:** Represents the implementation of an implicit component service and details of its realization, **Context:** Sets the environment in which the component service exists and where it can be adopted. **Choreography:** Specifies whether the service component is a composite service or not.

Du (Du et al., 2007) proposed another meta-model that includes contracts to the service component

Contracts can specify both the functional properties and the properties of QoS (Quality of Service) of the various system service components.

3 TECHNIQUES AND MODELS FOR QUALITY ASSURANCE OF COMPONENT-BASED SYSTEMS

As Service Component Architecture is a recent approach, there are very few research works for addressing quality issues and trustworthiness of service-component based systems.

In this section we present main techniques and models for quality assurance of component-based systems in general. We have identified two main classes, *a priori* and *a posteriori* approaches, the first one address quality issues in build and construction time and are rather considered *process-based* approaches. The second one is a product-based approach and is concerned by testing and correcting the final *product*.

3.1 Fault Tolerance

Anderson and Lee (Anderson and Lee, 1981) have identified four phases of fault tolerance: error detection, damage assessment, restoration of the state, and continuous service:

- **Error detection** determines the presence of a fault by detecting an erroneous state in a system component.
- **Damage assessment** determines the extent of damage to the system state caused by component failures and limit damage to the extent possible.
- **Restoration of the state** realizes the recovery of the error by restoring the system to a defined state, error-free.
- **Continuation of service**, despite the fault has been identified, means either that the fault must be repaired or the system must operate in a configuration where the effects of the fault no longer lead to an erroneous state.

Fault tolerance is a primary means of reliability of components-based systems. Mohamed and Zulkerine (Mohamed and Zulkerine, 2009) classified the efforts of fault tolerance depending on the type of component into three main classes: programming based paradigm, environment based fault tolerance models based fault tolerance.

3.2 Reliability Evaluation and Prediction

The evaluation and prediction of reliability is to predict the failure rate of components and overall system reliability. They can be used in the operational phase and the early stages of system design software.

Reliability models of the system are classified into three types: state-based, path-based and propagation-based.

- **State-based:** uses control flow graph to represent the system architecture and software to predict the reliability analytically.
- **Path-based:** software reliability calculated by considering the possible execution paths of the program
- **Propagation-based:** includes the dependence of component failures by considering the error propagation between system components

3.3 Certification

There is no consensus on the definition of components certification in component-based systems engineering, Councill (Councill and Heineman, 2001) proposed the following definition:

Third-party certification is a method to ensure software components conform to well-defined standards; based on this certification, trusted assemblies of components can be constructed.

Research on the quality of component-based systems can be divided in time into two categories: pre-2001 research focused on mathematical proofs and tests, and then the research has focused on models and prediction techniques and on quality assessment (Alvaro et al., 2005).

3.4 Quality Assurance Models

There are several standards for controlling and improving software and development process quality like ISO9001 and CMMI model (SEI), however, there is still no standard and effective process specific for component-based systems.

In the literature, several frameworks and quality assurance models have been proposed for the development of systems to bases of components like (Cai et al., 2000), (Meyer, 2003) and (Andreou and Tziakouris 2007).

A recent component quality framework is proposed Alvaro et al. (Alvaro et al., 2010), the framework consists of four modules: A **Component Quality Model** that determines which quality characteristics should be considered and which sub-attributes are necessary, **Evaluation Techniques Framework** which defines a number of techniques to be applied to software components evaluation, **Evaluation Process** in charge of defining a set of techniques, metrics, models and tools to assess and certify software components, to establish an evaluation standard for components and finally **Metrics Framework** responsible for defining a set of measures to monitor the properties of components and control the evaluation process.

3.5 Design by Contract

Design by Contract (Meyer, 1992) is an approach and method of software design. It is based on the legal definition of contracts which binds both parties and highlights the interest to precisely specify the interfaces behavior of a software component in terms of preconditions, post conditions and invariants.

The contract based approach provides proofs of non-functional and quality properties without requiring the full formality of proof-directed and mathematical development. This approach is particularly appropriate in the component-based context. In fact, a pre-condition on the parameters of an operation or a service defines a contract that the required/given component agrees to respect. Conversely, post-conditions on the return types of a required component define the customer's expectation from the service provider.

Beugnard (Beugnard et al., 1999) proposed a classification of contracts into four categories: **basic contracts** that ensure the possibility of running the system properly, **behavioral contracts** that improve trust in the system functionalities, **synchronization contracts** that specify synchronization strategies and policies, and finally **QoS contracts** which is the highest level and specify quality of service attributes.

Many contract extensions have been proposed for other programming and modeling languages like UML (Weis et al., 2001), (Warmer and Kleppe,

2003) Java and C#.

For component-based systems, Messabihi et al. (Messabihi et al., 2010) proposed multilevel contracts for components, however no complete contract feature has been proposed for service component until now.

It is important that quality is considered during all stages of the development lifecycle of the software. In fact, the contract-based approach allows both defining the desired quality properties and verifying and validating their accuracy.

4 OUR CONTRIBUTION

Dependability is a major requirement of the modern systems which consists of the system's ability to offer a trusted service.

To meet these requirements, we choose a contractual approach. Indeed, within the component and service paradigms, contracts have become a part of their definition (Szyperski, 1998):

A software component is a unit of composition with ***contractually*** *specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.*

A contract defines the constraints between components that is to say, the rights and obligations between the service provider and the client. It has the advantage of expressing the conditions of use of a service by clarifying the obligations and benefits of stakeholders.

Unlike mathematical evaluation and prediction techniques, the contract-based approach is a light-weight formal method for spefiying and designing quality-driven systems, it can be introduced in a early stage during the design phase.

To our knowledge, there is still no research work for introducing the concept of contract in service component based systems in order to manage and handle quality requirements.

Our future work will focus on proposing a framework for quality-driven Service Component Architecture by including, among others, the notion of contract for handling quality properties. The defined quality and safety contracts will become part of the design specification.

5 CONCLUSIONS

This work presented a literature review of main techniques and models for modelling and verifying quality-driven systems, we concluded that contract-based approach is very suitable for component-based systems in general and service component based systems in particular.

Contracts is a design approach for describing both functional and non-functional properties of complex and quality-driven systems, it also involves synchronization and Quality of Service (QoS) aspects.

As a continuation of this work, a framework should be presented and adapted to Service Component Architecture for safety-critical and quality sensitive systems.

REFERENCES

Alvaro, A., Almeida, E. S., Meira, S. R. L., 2010. A software component quality framework. In *ACM SIGSOFT Software Engineering Notes* 35, 1–18.

Alvaro, A., Almeida, E. S., Meira, S. R. L., 2005. A Software Component Certification: A Survey, In *the 31st IEEE EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, Component-Based Software Engineering Track.

Anderson, T., Lee, P., 1981. Fault Tolerance: Principles and Practice. Prentice-Hall, Englewood Cliffs, NJ.

Andreou, A. S., Tziakouris, M., 2007. A quality framework for developing and evaluating original software components. In *the Information & Software Technology*.

Barber, G., SCA Policy Framework Specification, http://www.osoa.org/display/Main/SCA+Policy+Framework.

Beisiegel, M., 2007, Service Component Architecture Specification, http://www.osoa.org/display/Main/Home.

Beugnard, A., Jezequel, J., Plouzeau, N., and Watkins, D. 1999. Making components contract aware. In *IEEE Computer* 32(7):38–45.

Cai, X., Lyu, M. R., Wong Roy Ko, K.-F., 2000. Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes. *International Journal of Software Engineering and Knowledge Engineering*.

Councill, B., Heineman, G. T., 2000. Component-Based Software Engineering and the Issue of Trust. *In Proceedings of the 22nd International Conference on Software Engineering*. ACM Press.

Ding, Z., Chen, Z., Liu, J., 2008. A Rigorous Model of Service Component Architecture. *Electr. Notes Theor. Comput.*

Du, D., Liu, J., Cao, H., 2008. A rigorous model of contract-based service component architecture. In *CSSE (2). IEEE Computer Society*.

Isaksen, U., Bowen, J. P., Nissanke, N., 1997. System and Software Safety in Critical Systems. *Technical Report RUCS/97/TR/062/A, Department of Computer Science, The University of Reading*, UK.

Messabihi, M., André, P., Attiogbé, C., 2010. Multilevel contracts for trusted components. *In Javier Camara, Carlos Canal, and Gwen Salaun*, editors, WCSI, volume 37 of EPTCS.

Meyer, B., 1992. Applying "design by contract". Computer.

Meyer, B., 2003. The Grand Challenge of Trusted Components. *In Proceedins ICSE 2003, IEEE Computer Society Press*.

Mohamed, A., Zulkernine, M., 2009. A Comparative Study on the Reliability Efforts in Component-Based Software Systems, Technical Report No. 2009-559, *School of Computing, Queen's University*, Kingston, Ontario, Canada.

Moore, R., Lopes, J., 1999. Paper templates. In TEMPLATE'06, *1st International Conference on Template Production*. SciTePress.

Papazoglou, M. P., Georgakopoulos, D., 2003. Service-Oriented Computing. *Communications of the ACM*.

Software Engineering Institute, CMMI site, available at www.sei.cmu.edu/cmmi.

Szyperski. C., 1998, Component Software: Beyond Object-Oriented Programming. *ACM Press and Addison-Wesley*, New York.

Warmer, J., Kleppe, A., 2003. The Object Constraint Language: Getting Your Models Ready for MDA, *2nd edition, Addison-Wesley*.

Weis, T., Becker, C., Geihs, K., Plouzeau, N., 2001. A UML Meta-model for Contract Aware Components, In *Proceedings of UML 2001*, Springer.

Zhang, L-J., Zhang, J., Design of Service Component Layer in SOA Reference Architecture, 2009. In *33rd Annual IEEE International Computer Software and Applications Conference*.