An Autonomic Service Oriented Architecture in Computational Engineering Framework

M. Agni Catur Bhakti and Azween B. Abdullah
Department of Computer and Information Sciences
Universiti Teknologi PETRONAS
Bandar Seri Iskandar 31750 Tronoh, Perak, Malaysia
E-mail: m.agni.cb@gmail.com

Abstract
Service Oriented Architecture (SOA) technology enables composition of large and complex computational units out of the available atomic services. Implementation of SOA brings about challenges which include service discovery, service interaction, service composition, robustness, quality of service, security, etc. These challenges are mainly due to the dynamic nature of SOA. SOA may often need to dynamically organizing (and re-organizing) its topologies of interactions between the services due to unpredictable events, such as crashes or network problems, which will cause service unavailability. The complexity and dynamism of the current global network system require architecture that capable of autonomously changing its structure and functionality to meet dynamic changes in the requirements and environment with little human intervention. In this paper, we elaborate the idea of adapting the autonomic computing paradigm into SOA in order to meet those challenges, and we present a case study of the idea in computational engineering framework.

Keywords: Autonomic computing, service oriented architecture.

1. Introduction

Development of internet technologies has enabled access to many types of distributed services over the web. Distributed systems, which provide services, resources, etc., are gaining an increasing importance and demand in our today’s lives. Using Service Oriented Architecture (SOA), large distributed computational units can be built on the base of existing services, by composing complex compound services out of simple atomic ones [1]. The visionary promise of SOA is to assemble application components with little effort into a network of services that can be loosely coupled and used to create flexible dynamic business processes and applications that may span organizational boundaries and computing platforms.

As the global network systems grow, the scale and complexity of current distributed systems are also increasing and showing high dynamism [2]. This complex and dynamic systems will require a more robust and autonomic SOA to keep up with the dynamic requirements and environment. This paper proposes an autonomic SOA, which is an SOA incorporating autonomic computing paradigm [3], to meet the challenges above in computational engineering framework. The rest of this paper is structured as follows: Section 2 reviews SOA and autonomic systems; Section 3 discusses the proposed autonomic SOA; Section 4 discusses the case study of the proposed architecture in
computational engineering framework; Section 5 presents summary of this paper and direction for future work.

2. Background & Literature Review

2.1. Service Oriented Architecture

Service oriented computing is an emerging computing paradigm that utilizes services as the basic constructs to support the development of rapid and easy composition of distributed applications – even in heterogeneous environments. Service oriented architecture is the main architectural concept in the field of service oriented computing. The Organization for the Advancement of Structured Information Standards (OASIS) defines SOA as the following: “A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains” [4].

In SOA, all functions, or services, are defined using a description language and have invoke-able, platform-independent interfaces that are called to perform business processes. Each service is the endpoint of a connection, which can be used to access the service, and each interaction is relatively independent of each and every other interaction.

Typically, web services are used as basis to implement a SOA. Web service is a software system designed to support interoperable machine-to-machine interaction over a network [5]. It refers to accessing services over the web [6]. A major focus of web services is to make functional building blocks accessible over standard internet protocols that are independent from platforms and programming languages. These services can be new applications or just wrapping around existing legacy systems that make them network-enabled. Figure 1 shows web services architectural model [7]. As a basis for SOA, web services models incorporate how web services are advertised, discovered, selected, and used. The architecture has three main parts: a provider, a consumer (requestor), and a registry. Providers publish or announce their services on registries using Web Service Description Language (WSDL), where consumers find – using Universal Description Discovery and Integration (UDDI), and then invoke them.

![Figure 1. Web services architectural model [7].](image-url)
2.2. Autonomic Computing Paradigm

The autonomic computing paradigm, inspired by the human autonomous nervous system, has been proposed as an approach for the development of computer and software systems that can manage themselves in accordance with (only) high-level guidance from human / administrators [3]. The autonomic systems consist of autonomic elements, whose behavior is controlled by autonomic manager.

Figure 2 shows the structure of autonomic element [3]. The autonomic manager will relieve humans of the responsibility of directly managing the managed element by monitoring the managed element and its external environment, and constructing and executing plans based on an analysis of this information (running the autonomic computing cycle, i.e. monitor, analyze, plan, and execute, using its knowledge base).

3. Autonomic SOA

Based on the aforementioned background, our idea is to adapt autonomic computing paradigm into SOA in order to achieve a more adaptive and robust SOA. In order to pursue the idea further, we have initiated our research whose goals are to provide new concepts of adapting autonomic, self-organizing mechanism into SOA, and to develop the functional SOA based on the proposed concepts. This work extends and elaborates the architecture in more detail than the previous work in [8].

Figure 2. Structure of autonomic element [3].

Figure 3 shows the overall architecture of an autonomic SOA that extends a typical SOA by incorporating autonomic computing paradigm into it. The architecture is separated to the three layers: the top is a presentation layer that provides interface for various users, the mid is a processing layer that performs and coordinates several jobs, and finally the bottom is a resource layer that enables to utilize the distributed resources via web services.
The resource / service layer is a typical SOA framework which consists of service providers, service registry, and broker agents (in processing layer) as service requestors. We extend the functionality of the service registry by adding knowledge base required by the autonomic processes. Knowledge base provides the capability to store the previous services profiles, which include: the type of service (atomic, composite), the atomic services that make up the composite service (the “ingredients”), where, when, and how to access them (the “recipe”).

The autonomic concept is incorporated in the processing layer where it has the autonomic manager. As in the autonomic computing paradigm, the autonomic manager will carry out the autonomic cycle as the following:

- Monitoring the system which will include monitoring of the requests from users, the availability of the services, addition of new services, removal of services, etc. A sentinel or monitoring agent will provide the monitoring service.
- Analyzes incoming service requests. It will retrieve service profile information from knowledge base and will use and / or revise the information as necessary to provide the requested service. It will search for information of the requested service in knowledge base and the options are the following:
  - If the information of that specific service is available in the knowledge base, it will reuse that information to plan actions.
  - If there is no information of that specific service in the knowledge base, the system will search for similar ones, and then it will reuse and/or revise as necessary.
  - If there are no similar previous cases, the monitoring agent will search for the composite service in service registry (or search for atomic services that could be composed into the requested service).
The new revised information then will be used to create new action plan and updated to the knowledge base.

- Plans suitable actions to provide the requested service. If the requested service is a composite service, the action plan will include the list of available atomic services that will be composed into the composite service, where and how to access them, and the sequence of accessing them. It will also update the knowledge base if new action plan is created, so that it can be used when the same composite service is requested again in the future.

Here, we adopt the Case-Based Reasoning (CBR) cycle [9], i.e. retrieve, reuse, revise, and retain, for adaptive and learning features which will include the analysis and planning processes above and knowledge base. CBR is chosen because it has been successfully adapted and applied into other autonomic system, i.e. self-healing system in [10].

- Executes the plan to come up with the requested service. Broker or negotiator agent will assist in interacting and negotiating with the applications to obtain the required services. The composer / aggregator compose the atomic services to provide composite service.

The autonomic manager will also suggest other services to the users which are related to the requested services (e.g. other services that are also typically used) based on the previous cases in the knowledge base.

4. Case Study in Computational Engineering Framework

Engineers and researchers in computational engineering typically use different types of software and applications in its workflow. The applications might include modeling, simulation, and visualization software. The software environments sometimes are complex and dynamic, supporting different software packages, codes, and possibly distributed in different locations. The use of SOA will help integrating the different applications / software used in computational engineering.

Figure 4 illustrates an on-going project that has been initiated aiming to provide an integrated services framework for various distributed software used in computational engineering research, in-house software and commercial software, including modeling software, simulation software, and visualization software. We are currently implementing the proposed architecture in this project using Java and Apache Axis2 as the Web Services / SOAP / WSDL engine. The modeling software will include computational software using Ray tracing technique and Finite-difference Time-domain (FDTD) method. The simulation software will run on High Performance Computing (HPC) platform. The visualization software will include 2D visualization tools (graphs and charts), 3D visualization tools, and virtual reality visualization tools.
In engineering research, we cannot fix which implementation software we want to use. For example, right now, the engineers may be using a commercial FDTD version. But later, they may switch to in-house built version, and a couple weeks later to a freeware version. Over the course of work, the autonomic feature will be able to suggest the users (engineers) which software to use and engineers will develop a sense of which software implementation is most suitable for which specific purpose. Figure 5 shows an instance of computational engineering process choreography. For example, the system could use ray-tracing and FDTD modeling software, and virtual reality software for visualization. We can see that some actions are initiated by user and other actions are initiated by the autonomic manager automatically. The autonomic manager will learn and adapt appropriately the sequences of tasks to solve computational engineering problems based on the previous cases stored in knowledge base.

Figure 4. Implementation model of SOA in computational engineering framework.

Figure 5. Example of interaction model in computational engineering framework.
5. Summary & Future Work

This paper presented the idea of applying autonomic computing paradigm into SOA in computational engineering framework. We have designed the architecture of the proposed autonomic SOA and its implementation model in computational engineering framework. It is promising that the autonomic paradigm would improve the usability, adaptability, and robustness of SOA.

Currently we are developing a prototype of the proposed system for the mentioned project / case study. SOA framework, autonomic paradigm, web services, workflow, and business process modeling are the main technologies used in the project. When the current framework is developed successfully in near future, it would provide a more effective and efficient research and also savings of times and costs in the fields of computational engineering. And it is expected the system will be able to be extended for other applications in various domain.

References
