Chapter 2
Dynamically Reconfigurable Architectures:
An Evaluation of Approaches for Preventing Architectural Violations

Marek Rychly
Brno University of Technology, Czech Republic

ABSTRACT

Dynamic aspects of behavior of software systems in dynamically reconfigurable runtime architectures can result in significant architectural violations during runtime. In such cases, a system's architecture evolves during the runtime according to the actual state of the system's environment, and consequently, runtime reconfigurations may eventually lead to incorrect architecture configurations that were not considered during the system's design phases. These architectural violations are known as architectural erosion or architectural drift, and they contribute to an increasing brittleness of the system, or a lack of its coherence and clarity of its form. This chapter describes and compares possible measures to prevent architectural violations in dynamic service and component models. The aim of this chapter is to evaluate the applicability of those measures in combination with advanced features of reconfigurable runtime architectures such as ad hoc reconfiguration, service or component mobility, composition hierarchy preservation, and architectural aspects.

INTRODUCTION

Current information systems tend to be designed as component-based systems and often utilize Service Oriented Architecture (SOA) and Web service technology. The service orientation enables decomposition of a complex software system into a collection of cooperating and autonomous components known as services. These services cooperate with each other to provide a particular functionality of the implemented software system with defined quality.

Loose binding between the services, which represent individual components of a system, enables *runtime reconfigurations* of the system architectures. In other words, it enables creating, destroying, and updating the services, and establishing and destroying their interconnections dynamically at runtime, on demand, and according to various aspects to move the services into different contexts and to different providers (i.e., service mobility). Eventually, a series of reconfigurations contributing to the evolution of the architecture, of a supposedly well-designed system may lead to incorrect architecture configurations that were not considered during the system's design phase. These incorrect configurations are commonly known as *architectural violations*.

This chapter describes and compares possible measures to prevent the architectural violations, as they are used in the current state-of-the-art approaches. The goal is to evaluate applicability of those measures in combination with the advanced features of dynamic architecture such as ad hoc reconfiguration, service or component mobility, composition hierarchy preservation, and architectural aspects.[1] Specific objectives include an introduction to the problems of dynamically reconfigurable runtime architectures, an analysis of the state-of-the-art approaches in this field

with focus on the advanced features of dynamic architectures, and the methods to prevent architectural violations.

The chapter is organized as follows. The next section deals with software architecture in general and introduces component-based development and service-oriented architecture with concepts of dynamically reconfigurable runtime architectures. We also describe several important state-of-the-art works dealing with component-based development and component models supporting features of dynamic and mobile architectures. In the following section, we discuss existing problems relating to the support of dynamic and mobile architectures that cause architectural violations in component-based or service-oriented systems.

Then, we outline possible strategic improvements and introduce approaches to prevent the architectural violations in general, and also describe their applications in the current state-of-the-art related works. The next part of the chapter deals with the evaluation of the previously described approaches for preventing architectural violations. More specifically, we analyze compatibility of the approaches with the advanced features of dynamically reconfigurable runtime architectures. Finally, we discuss future research directions such as possibilities of utilization of the advanced features of dynamically reconfigurable runtime architectures including previously described methods of preventing architectural violations in implementations of service-oriented architectures.

BACKGROUND

According to IEEE (2000), software architecture is defined as the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and

16 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage: www.igi-global.com/chapter/dynamically-reconfigurable-architectures/115422

Related Content

Traceability in Model-Driven Software Development
Ståle Walderhaug, Erlend Stav, Ulrik Johansenand Gøran K. Olsen (2009). *Designing Software-Intensive Systems: Methods and Principles (pp. 133-159).*
www.irma-international.org/chapter/traceability-model-driven-software-development/8236

Weaving Security into DevOps Practices in Highly Regulated Environments
Jose Andre Morales, Hasan Yasarand Aaron Volkmann (2018). *International Journal of Systems and Software Security and Protection (pp. 18-46).*
www.irma-international.org/article/weaving-security-into-devops-practices-in-highly-regulated-environments/221157

Request and Response Analysis Framework for Mitigating Clickjacking Attacks
Hossain Shahriar, Hisham Haddadand Vamshee Krishna Devendran (2015). *International Journal of Secure Software Engineering (pp. 1-25).*
www.irma-international.org/article/request-and-response-analysis-framework-for-mitigating-clickjacking-attacks/136450

Person Identification Using Top-View Image with Depth Information
Daichi Kouno, Kazutaka Shimadaand Tsutomu Endo (2013). *International Journal of Software Innovation (pp. 67-79).*
www.irma-international.org/article/person-identification-using-top-view-image-with-depth-information/89776

CbSSDF: A Two-Layer Conceptual Graph Approach to Web Services Description and Composition – A Scenario Based Solution Analysis and Comparison with OWL-S
Xiaofeng Du, Malcolm Munroand William Song (2011). *International Journal of Information System Modeling and Design (pp. 82-103).*
www.irma-international.org/article/cbssdf-two-layer-conceptual-graph/58646