A Programmble Fault Injection Testbed Generator for SOA

Lukasz Juszczyk and Schahram Dustdar

Distributed Systems Group, Vienna University of Technology, Austria
{juszczyk,dustdar}@infosys.tuwien.ac.at

Abstract. In this demo paper we present the prototype of our fault injection testbed generator. Our tool empowers engineers to generate emulated SOA environments and to program fault injection behavior on diverse levels: at the network layer, at the execution level, and at the message layer. Engineers can specify the structure of testbeds via scripts, attach fault models, and generate running testbed instances from that. As a result, our tool facilitates the setup of customizable testbeds for evaluating fault-handling mechanisms of SOA-based systems.

1 Motivation

Today service-oriented architectures (SOAs) do not only comprise Web services, clients, and brokers, but also diverse complex components that operate on distributed service-based environments, e.g., workflow engines executing business processes or governance systems managing whole large-scale systems. Usually, these components carry out critical tasks and, therefore, must be dependable. However, if one considers that SOA-based infrastructures often have complex dependency structures and each service is a potential fault source, it becomes obvious that single faults can trigger chains of harmful effects. Consequently, complex SOA components must provide fault handling mechanisms in order to guarantee a certain level of dependability.

In our current research, we regard this challenge from the perspective of how such mechanisms can be evaluated in order to verify their correct execution. We argue that this can only be achieved by testing the developed component at runtime, in an environment which comprises faulty elements, and by observing the component's reaction on occurring faults. Yet, the main obstacle remains how to get access to such an environment which exhibits faulty behavior and, therefore, can be used as a testbed infrastructure. As a contribution to solving this issue we have developed the *Genesis2* testbed generator framework and extended it with fault emulation techniques. Our tool provides SOA engineers the facility to generate customizable testbeds for evaluating their complex components.

2 Fault Injection Testbed Generator Framework

The presented prototype uses the *Genesis2* testbed generator framework [1] (in short, G2) as a base grounding for emulating testbeds for SOA. Engineers model

P.P. Maglio et al. (Eds.): ICSOC 2010, LNCS 6470, pp. 722–723, 2010.

the composition of required testbeds via a *Groovy*-based [2] scripting language, define the communication topology among the testbed's components, and program the functional behavior of these. G2 interprets these models, generates real SOA components, and deploys them on a distributed back-end environment, creating this way testbed infrastructures on-demand. For the sake of extensibility G2 comprises a modular architecture and applies composable plugins that extend the framework's functionality and augment the generated testbeds, for instance by providing the generation of additional SOA artifacts.

Based on G2 framework we have developed techniques for injecting multi-level faults into running testbeds in order to emulate typical failures in service-oriented computing. In the current state we support the following fault types:

- faults at the *message layer*, in terms of message data corruption,
- faults in *service execution*, for simulating quality of service (QoS) issues, and
- faults at the *network layer*, hampering the IP packet flow between hosts.

A distinct feature of our tool is the ability to program fault models via the scripting language. It provides fine grained control on the fault injection mechanisms and allows engineers to customize the functionality of their testbeds to their requirements of the tested complex SOA component.

For a mode detailed description of how we perform fault injection we refer readers to [3] and for learning how G2 generates testbeds we recommend to read [1].

3 Prototype Demonstration

In the demo we will present the prototype implementation of our testbed generator. Via sample scripts we will demonstrate how testbeds are modeled, how faulty components are specified, and, eventually, how faults are injected into running testbeds. Particular focus will be put on the programmability of the fault models, which is the most novel contribution of our approach. We will show the effects of injected faults on a deployed complex SOA component (e.g., a BPEL workflow engine) by visualizing the intercepted communication and pointing out occurring misbehavior.

The provision of our prototype as open source software (downloadable at [4]) will be an additional asset of our demonstration.

References

1. Juszczyk, L., Dustdar, S.: Script-based generation of dynamic testbeds for soa. In: ICWS, pp. 195–202. IEEE Computer Society, Los Alamitos (2010)
2. Groovy Programming Language, `http://groovy.codehaus.org/`
3. Juszczyk, L., Dustdar, S.: Programmable Fault Injection Testbeds for Complex SOA. In: ICSOC. LNCS, Springer, Heidelberg (2010)
4. Genesis Web site, `http://www.infosys.tuwien.ac.at/prototype/Genesis/`