Novel Collaborative Automated Testing Framework Using DDF*

Songwen Pei, Baifeng Wu, Qiang Yu, and Kun Zhu

Department of computing and information technology, Fudan University,
Shanghai 200433, P.R. China
061021045@fudan.edu.cn

Abstract. Collaborative testing is an effective way of distributed interoperability in pursuit of automated testing. In this paper, a novel collaborative testing approach named Collaborative Automated Testing Framework (CATF) which meets the requirements of not only automated testing but also collaborative operation is proposed. Through the abstract analysis in terms of extended dynamic dataflow (DDF) model's viewpoint incorporating with UML2.0 profile of MDA, we design the framework with an automated engine working as a Finite State Machine (FSM). Particularly, as a approach to collaborative testing at a system level, CATF is implemented with component modules based on J2EE and verified to be of efficiency.

1 Introduction

Collaborative working on software is a critical and perpetual issue. As a consequence, the issues about collaborative developing and collaborative testing are surging out. In particular, automated and collaborative testing oriented to distributed systems and its applications is essential in effective development of reliable software. It is a hotspot not only in the domain of industrial application but also in the research field focusing on computer aided design, software development, etc. [1, 2]. Unfortunately, the dramatic expansion of software complexity and the shorter time-to-market become the strongest pressure and bring critical challenges to the research and industrial communities.

There has been lots of research referring to collaborative testing. However, scarcely has any approach to automated testing system-level model involved abstract dataflow model so far. Hence, we focus on this issue with an improved and integrated abstract testing approach based on dynamic dataflow (DDF) not only catering for collaborative operations but also meeting automated testing requirements. The separation of various aspects of concerns allows more effective exploration of alternative solutions, particularly the separation of function from architecture and the extraction of dataflow and control flow from system. In the Collaborative Automated Testing Framework (CATF), an extended dynamic dataflow corresponding to partitioned role relationships is considered, and a system-level Finite State Machine (FSM) is adopted in automated testing engine so as to guarantee a dynamic balance between input and output dataflow in the whole of testing system. All the testing operations through web interface are

Y. Luo (Ed.): CDVE 2007, LNCS 4674, pp. 385–395, 2007.

decompounded by DDF model and the role compatibility of various operations is guaranteed by the role interface algorithm.

Under the guide of the concept of dynamic dataflow, we have implemented the automated testing engine with distributed functions aiming at collaborative and concurrent testing using J2EE [3] model. The testing framework is modeled in terms of Model-View-Control (MVC) principle. It consists of five modules as a whole: Web Interface, Queue Pool, Dynamic Scheduler, Resource Manager Module and Test Result Module. The Web Interface module is a part of View in the MVC, and Queue Pool and Dynamic Scheduler model is corresponding to the Control in the MVC. Besides, the Execution Server in test engine is implemented by java beans or enterprise java beans to accomplish the tasks of Model in the MVC. Among others, the Resource Manager and Test Result module are supplements for advancing the efficiency and the quality of testing and fixing the bugs of tested functions.

The rest of this paper is organized as follows: In Section2, we review the related work about collaborative testing. The abstract analysis based on DDF will be discussed in Section 3. In Section 4, the critical models of CATF that are job scheduler and state transfer of FSM are expatiated on respectively. The novel collaborative automated test framework (CATF) is proposed in Section 5. At last, we conclude in the final section.

2 Related Work

The typical approach to depicting collaborative working depends on the Collaboration State Chart of UML. Therefore, modeling software system with UML testing profile is a predominant approach in the domain of collaborative working currently [4]. The main idea of Model Driven Architecture (MDA) is using UML to specify both the static interfaces and the dynamic behavior of the components in Platform-Independent Models (PIMs). Additionally, it defines rules so that PIMs can be mapped into a number of Platform-Specific Models (PSMs). Therefore, the MDA strengthens the concepts of portability and interoperability due to the transformation between PIMs and PSMs. Raul Silaghi, et al. [5] introduced the abstract and distributed realization profile oriented to MDA, but they didn't solve the collaboration issues existing in the distributed system. TPTP-based tools are not only interoperable but also tightly integrated with the Eclipse Platform-based development tools [6], but it can't accomplish automated testing directly. Recently, Pyxis Technologies has launched their testing product GreenPepper which is a platform intended to improve collaboration between business experts and software developers [7], but it cannot support automated testing among several testers. Petrenko, A.[8]proposed a queued-quiescence testing framework based on input-output transition systems, and it applied inputs via a queue to an implementation under test (IUT) and detected outputs from IUT. However, it lacks an effective mechanism to detect deadlocks occurred in queue. Jan Tretmans et al. [9, 10] designed an automated model based testing called TorX which is based on the ioco-test theory. In contrast with the framework of Torx, we intend to execute test cases and test suite automatically and collaboratively. However, they focus on how to generate test case and suite automatically.

Starting from a high-level of abstract about collaborative testing, with the capability of partitioning roles compatibly and decompounding functions properly using extended