Providing Software Engineering Services for Virtual Software Organizations (Work in Progress)

Frank Maurer, Jeremiah Wittevrongel
The University of Calgary
maurer@cpsc.ucalgary.ca, jeremiah@cpsc.ucalgary.ca

Abstract

In this abstract, we briefly sketch the idea of providing a software engineering service on the Web instead of selling SE tools to customers. We illustrate this approach by our scenario-based testing tools, SCENTOR.

1. Introduction

The development of e-Business software faces extreme challenges by fast changing technologies. Hence, it is often not cost effective for small development organizations to buy and install the latest version of specific tools locally. On the other side, using old versions of these tools is also problematic because newer versions may provide useful features for the current software development project. This problem is increasing in case of virtual software organizations whose members cooperate on one specific project without being able to integrate their processes and tools to a common base.

An alternative to buying a tool is leasing a service for a limited time. This reduces installation and maintenance effort but, nevertheless, allows using current versions whenever required.

One example of an advanced SE service is scenario-based testing of Java applications. As a proof of concepts, we developed a web-based tool, SCENTOR, which is able to provide this service for a virtual team working on EJB-based software [9].

SCENTOR [12] is an approach that aims to provide e-business-specific support for the generation of scenario-based tests using JUnit [7] as a basis. It is accessible over the Web without local installation.

E-business projects often have huge time-to-market pressure; there is not always a lot of time for testing. Priority One under extreme time pressure should be ensuring typical use scenarios can be completed.

In Extreme Programming [1,2], this user-visible functionality is described in user stories. SCENTOR uses Unified Modeling Language (UML) sequence diagrams on top of this approach, which serves to make the scenarios slightly more formal to allow useful support and partial automation for the tedious task of writing automated test drivers. The developer would only need to add concrete parameters and expected results for each step in the scenario, and this forms a straightforward automated test driver.

2. From scenarios to test drivers

SCENTOR is targeted towards lightweight development processes that include only a partial set of UML models while maintaining the Extreme Programming focus on the production of source code.

Scenarios are first modeled as UML sequence diagrams, using a CASE tool such as Rational Rose. This includes classes and methods participating in the scenario. The UML model is then exported from the CASE Tool in the vendor-independent XMI (XML Metadata Interchange) format.

The developer would then load the XMI file into SCENTOR, and could optionally load a previously created test specification (in an XML format [11]) as well.

Tests are specified based on the UML sequence diagrams. A small suite of tests is typically based on the set of messages sent by a single object in a UML sequence diagram. This single object could, for example, represent the whole user interface of the system. The developer needs only to add concrete parameter values to the method call and specify the expected results of the method call. To preserve the focus on source code and avoid the need for developers to learn another language, we decided to let the developer enter these results in the underlying programming language (Java) instead of using other formal languages (e.g. UML's object constraint language, OCL).

SCENTOR concretely helps in the development of test drivers for Enterprise Java Beans by supporting generation of common EJB-specific setup code. This setup code would normally be shared among a group of test cases in such a way that it is only executed once, regardless of the number of tests running. Creating the initial context for Bean lookups and retrieving references to the EJB Home Interfaces are two tasks that are well suited for this type of setup code; SCENTOR supports both.

Developers can compile and execute the generated test drivers from within SCENTOR if they wish, or may employ another compilation and execution environment.

The Test specifications can be saved as an XML files following the SCENTOR Test Specification DTD [11]; these files can later be loaded into SCENTOR for modifications or additions.

3. Summary and future work

SCENTOR assists developers by forming a bridge between user scenarios and functional test drivers. By removing some of the repetitive, mechanical work required when creating automated test drivers, SCENTOR also aims to reduce the time required to develop them.

Furthermore, SCENTOR provides a free software engineering service on the Web and allows us to find out if its intended user community accepts such a service. This can then be a basis for determining and evaluating potential pricing models for a commercial services in the same or in other areas.

8. References

[1] K. Beck, "Embracing Change with Extreme Programming", *Computer*, IEEE CS Press, Los Alamitos, Calif., vol. 32, no. 10, Oct. 1999, pp. 70-77.
[2] K. Beck, *Extreme Programming Explained: Embrace Change,* Addison Wesley, Reading, Mass., 1999.
[3] L. C. Briand, *Testing of Object-oriented software sysTEms with the Unified Modeling Language (UML),* http://www.sce.carleton.ca/faculty/briand/totem/totem.htm (current Feb. 10, 2001).
[4] M. Fewster and D. Graham, *Software Test Automation: Effective use of Test Execution Tools,* Addison Wesley, Harlow, England, 1999.
[5] M. R. Lyu, ed., *Handbook of Software Reliability Engineering,* IEEE CS Press, Los Alamitos, Calif., 1996.
[6] B. Marick, *The Craft of Software Testing,* Prentice Hall, Englewood Cliffs, N.J., 1995.
[7] Object Mentor, Inc., *JUnit, Testing Resources for Extreme Programming,* http://www.junit.org/ (current 10 Feb. 2001).
[8] W. E. Perry, *Effective Methods for Software Testing,* Second ed., John Wiley & Sons, New York, 2000.
[9] Sun Microsystems, *Enterprise JavaBeansTechnology*, http://java.sun.com/products/ejb (current 23 Feb., 2001).
[10] J. D. Wells, *Extreme Programming: A Gentle Introduction,* http://www.extremeprogramming.org (current 10 Feb. 2001).
[11] J. Wittevrongel, *The SCENTOR Test Spec. DTD*, http://sern.ucalgary.ca/~milos/dtd/scentorTestSpec.dtd (current 23 Feb. 2001).
[12] J. Wittevrongel, *The SCENTOR Web Site*, http://sern.ucalgary.ca/~milos/projects/scentor_intro.htm (current 10 Feb. 2001).
[13] L. Wong, *The EBOLA Web Site,* http://sern.ucalgary.ca/~milos/projects/ebola/ (current 10 Feb., 2001).
[14] K. Zallar, "Practical Experience in Automated Testing," *Methods and Tools*, Martinig and Associates, vol. 8, no. 1, Spring 2000, pp. 2-9.