Automated Test Generation

(From a Behavioral Model)

James M. Clarke
Lucent Technologies
2000 Naperville Road
Naperville, IL 60666-7033
(630) 979-1861
jmclarke@lucent.com

Abstract

The challenge for testers: reduce the testing interval without reducing quality. One answer: find a new way to approach test design and test generation. This paper will discuss an ongoing Lucent Technologies experiment in automated test generation from a behavioral model of the software product under test. Results indicate that our new approach can increase the effectiveness of our testing while reducing the cost of test design and generation.

Outline

1. Introduction
2. 5ESS®-2000 Testing Background
3. Major Challenges for Testers
4. New Test Design Strategy – Behavioral Modeling
5. Automatic Test Generation
6. Case Studies
   - Case 1: Call Management Feature
   - Case 2: Number Portability Feature
7. Observations and Conclusions

1. Introduction

At Lucent Technologies, TestMaster™ automates the generation of tests for call processing features developed for the 5ESS®-2000 Switch. The 5ESS-2000 Switch, a digital exchange for use in the global switching network, allows service providers, such as telephone companies, to route ISDN voice and data, local voice calls, long distance calls, Internet

access, wireless PCS, Advanced Intelligent Network services, interactive video and multimedia services in a high-speed, reliable public network. During the test development phase, a call processing feature's specification document (FSD) serves as the basis for a TestMaster state-based model. The model describes the behavior of the switch/network when a call uses the associated feature. TestMaster then performs a path analysis on the model, generating a comprehensive set of tests that are formatted and executed in the Lucent 5ESS-2000 testing environment.

In this paper I will review the testing problems we faced, the solutions we found and the results of implementing those solutions.

2. 5ESS®- 2000 Testing Background

The 5ESS-2000 Switch is a flexible digital exchange for use in the global switching network. Digital switches replaced earlier electromechanical and analog switching systems. A digital switch is a single system with multiple applications such as local, toll, and operator services. The 5ESS equipment switches ISDN voice and data, local voice calls, long distance calls, Advanced Intelligent Network services as well as other media on the public switched network. The switch architecture is a modular, distributed architecture that allows developers to implement enhancements easily and allows service providers to change their communication network quickly.

The modular design of the 5ESS-2000 Switch also carries through to its software architecture. The software, primarily written in the C programming language, extends the many advantages of a distributed processing environment. Lucent Technologies Bell Laboratories develops and tests the software for the 5ESS-2000 Switches that FCC-required quality monitoring has shown to be four times more reliable than its nearest competitor.

At one time Lucent Technologies (at the time a business unit of AT&T) viewed testing as a standalone phase in the traditional waterfall process. System testing was done by a separate organization, and the testers became involved in a project only after the specifications, design, and the majority of the coding was complete. This made for expensive and time consuming test plans. In fact, at one time it required almost 22 months to deliver a major software release for the 5ESS-2000 Switch. Process and organizational changes have reduced that figure to approximately 10 months, but as new features become more complex, it has become increasingly difficult to maintain both an aggressive delivery schedule and the high level of software quality that our customers have come to expect.

3. Major Challenges for Testers

The challenge now for test plan designers is to continue to achieve the high degree of testing coverage required to ensure that these increasingly complex features maintain quality standards. This requires the use of test development methods that are more

effective in managing the coverage of complex functionality. Traditional methods, such as analyzing each requirement and developing test cases to verify correct implementation, are not effective in understanding the software's overall complex behavior. Also the cost pressures in a competitive industry add the constant of cost reduction. This adds the need for efficiency in using more effective test development methods. While initially these two goals, reduced testing costs and maintaining product quality, appeared to be mutually exclusive our automation test generation initiatives have indicated that this is not necessarily the case.

A Feature Specification Document, written by the systems engineering organization, details the requirements for the behavior of call processing features of the 5ESS. The behavior of the feature depends on inputs from the parties on the call and the configuration and signaling input from the 5ESS network. Complex interactions arise between the calling parties, other features on the switch, and the network, and these must be understood to adequately test the new feature. To date, test generation has relied on manual methods to interpret the Feature Specification Document, state diagrams, and call processing behavior of the switch. For a given call, the switch waits for input, e.g., a set of DTMF tones. The switch processes the input and changes the state of the call in progress. (Different inputs from the caller and network configurations cause the 5ESS switch to process calls differently.) For example, if the user enters a valid telephone number, the call will be processed; if not, an announcement will play asking for a valid input. Advanced features in the 5ESS switch have so many variables that it is difficult for the test engineer to identify them all, let alone generate a set of tests to verify that the feature works in all cases.

4. New Test Design Strategy – Behavioral Modeling

The traditional test design methods used to generate test cases became too expensive and labor intensive when applied to these highly complex features. We had to employ a different strategy to adequately test new and existing functionality, while keeping the testing interval from growing with the software complexity.

For the last 2 years our strategy has been to use requirements behavioral modeling on a number of features to determine the effectiveness of this approach as a test design and generation strategy. The behavioral modeling we use combines transaction-flow, control-flow and finite state machine (FSM) testing techniques in an extended finite state (EFSM) model. EFSM modeling employs a technique known as *predicate notation* to simplify models of complex systems, and reduce the state explosion problem commonly encountered with pure FSM modeling.
The goal was to create a EFSM model that would capture the functional behavior of the requirements for a new 5ESS-2000 software feature. The models, if created during the requirements definition phase of the development cycle, would prevent different interpretations of the requirements by the developers and testers. Minimizing these differences will in itself prevent some faults from ever reaching the test execution phase, helping to further reduce testing costs.

Our initial results indicate that while behavioral modeling is very effective in ensuring adequate coverage during the test design phase and in providing the entire development team with a common view of the requirements, it quickly becomes labor intensive during the test generation phase. On larger features the process of manually modeling also quickly becomes too difficult and expensive. An obvious answer was to find a tool that could automate some or all of this process. Which automation tool to use was not as obvious.

5. Automating Test Generation Using Model Reference Technology

To help decide which tool to use, we developed a checklist of the characteristics and functionality to rate automation tools- characteristics such as execution environment independence, support for EFSM, flexible output format and the ability to automatically generate unique paths (tests) from the behavioral model.

The tool with the best score based on our checklist is a product called TestMaster, and Lucent Technologies started a trial program with to evaluate its ability to allow test engineers to create and maintain behavioral models of our products.

TestMaster (produced by Teradyne, Inc.) uses model reference technology (MRT) to provide automatic test generation driven from an EFSM model of the application under test. TestMaster comprises three major components: a graphical editing tool, a test program generator, and a model debugger.

Using the same inputs used to manually generate test scripts or manually create an EFSM, test engineers use the State Transition Editor to build a model of the applications behavior. The model is a series of states connected by transitions. Each transition defines a state change based on inputs from user or switch. Each transition in the model contains the following associated programmable fields: the predicate and constraint fields, which evaluate context in the model, and the test information field that contains procedures or test code that will be included in any test case that includes the transition as part of its path. Predicates are boolean expressions that must evaluate *true* in order for the transition to be a valid path within the behavioral model. The constraint field allows the user to limit the number of paths produced during test generation. A set of interactive debugging tools is available to the test engineer as well.

The test program generator uses the model to automatically find valid paths through the model. These paths consist of transitions that represent the behavior of the application that has been modeled. Each valid path through the model is converted into a test case by replacing each transition in the path with its test information. Thus a complete test case is concatenation of all the test information field for some valid path. These test cases can be produced in any target language.

6. Case Studies

This section will briefly discuss two cases in which we can compare generating tests with TestMaster to manually writing tests. In both cases the two methods were used to create comparable type and number of test cases. Both of these cases are products that are currently available on the 5ESS-2000 Switch.

Case 1: Call Management Feature

Background

This feature expands the capabilities of basic Call Waiting to include a number of call management features. If you subscribe to Call Waiting on your analog phone line, and a third party calls you while you are on a phone call, you receive tones indicating that another call has arrived. At this point you only have two choices: press the phone switch-hook and answer the new call or ignore the new call.

Call Management provides the ability to see the new call's telephone number[1] and the name of the caller[2]. At this point you can conference the two calls together, place either call on hold (music optional), or forward the new call manually or automatically to another telephone number.

Test Generation: Manual vs. TestMaster

This product was delivered in two phases. Phase two testing required, modifying some of phase one's tests, using some of phase one's tests as is, and writing new tests. Test generation is measured by the Technical Head Count Year (THCY) effort required to produce the test cases required. For example, if the test generation took an engineer one month to complete, it would equal a 0.0833 THCY effort.

To generate the tests manually, we used traditional 5ESS-2000 call processing test design and generation methods. Then, using TestMaster, we created an EFSM for the product and automatically generated test cases. Table 1 compares the THCY effort required by these two methods for this feature.

	Manual Generation	TestMaster Generation
Phase One	0.120	0.014
Phase Two	0.050	0.002
Total	0.170	0.016

Table 1

[1] Caller ID Feature
[2] Calling Name Feature

The use of TestMaster in this case provided a test generation productivity improvement of just over 90%. At this level of test generation productivity improvement one test engineer using TestMaster can be as productive as ten test engineers using manual test generation.

Case 2: Number Portability Feature

Background

The competition to provide local phone service is increasing every year. But most people would probably decline to change their local service providers if changing companies meant changing phone numbers. The Number Portability (NP) feature, mandated by the FCC to overcome this barrier, allows you to switch service providers without changing your telephone number.

Test Generation: Manual vs. TestMaster

For this feature we manually created an EFSM behavioral model of the requirements and manually generated test cases using the model. Then, we created an EFSM of the product in TestMaster and automatically generated test cases. Table 2 compares the THCY effort required by these two methods for this feature.

Test generation is again measured by the Technical Head Count Year (THCY) effort required to produce the test cases required.

	Manual	TestMaster
Create Model	0.21	0.05
Generate Tests	0.24	0.00[3]
Total	0.45	0.05

Table 2

In this case TestMaster provided a test generation productivity improvement of just over 88%. Additional functionality was added to this feature after the original feature was released. Editing the TestMaster model to create the new tests case took half a day[4] compared to the estimate of two and a half weeks[5] for manual generation.

[3] Automated test generation, no THCY effort required.
[4] 0.00192 THCY effort
[5] 0.0288 THCY effort

7. Observations and Conclusions

TestMaster provides a single environment to capture the behavior, input variables, configuration, and 5ESS state information in the form of a model. TestMaster can then automatically process the model to quickly generate a complete set of tests. This technology provides the 5ESS-2000 call processing test team an efficient and effective method of generating feature tests for 5ESS development projects.

Since starting with TestMaster in September of 1996, we have successfully modeled, generated, and executed test cases for a number of advanced call processing features in the 5ESS switch. To increase the reusability of the models, test engineers are developing standardized methods for analyzing the Feature Specification Document and creating TestMaster models. We are also investigating ways to formally review the models for completeness. The test cases generated are in a standard format so they can be used by both manual and automated test executors who have no knowledge of the TestMaster model and who run the tests as if they were generated manually. We can easily incorporate changes into the models to keep pace with changing feature requirements.

Preliminary data indicates that using TestMaster to automate our test generation process can increase our productivity by over 80%, while providing a more effective way to analyze complex requirements.

