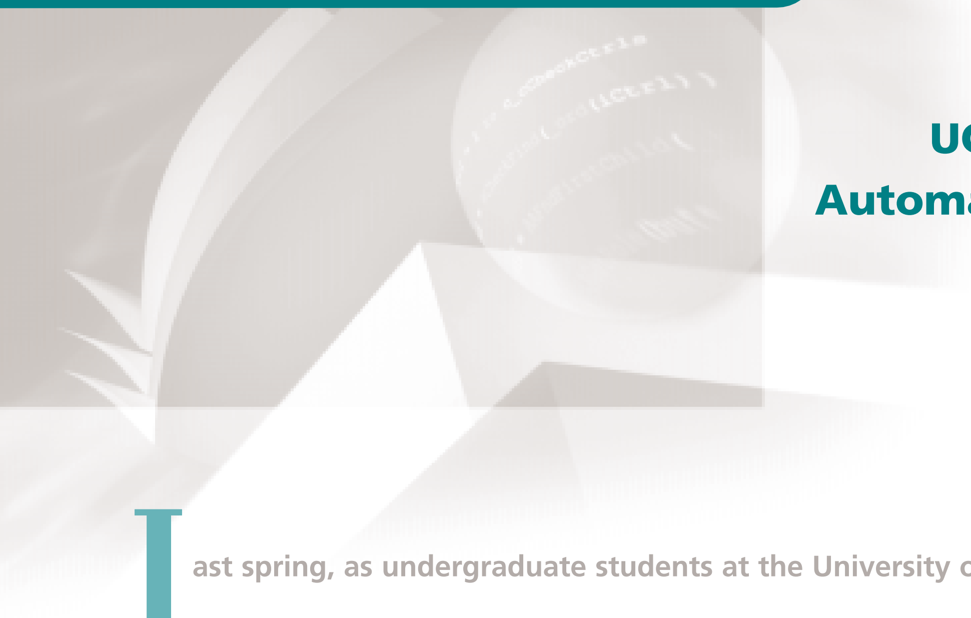
A CASE STUDY



UC Irvine Teaches Automated Testing Principles with Rational Visual Test

Last spring, as undergraduate students at the University of California, Irvine (UCI) pored over the course catalog to select their schedules, they came across a rare and exciting opportunity.

The Department of Information and Computer Science (ICS) was offering a course titled *Automated Software Testing Using Visual Test.* The course is believed to be the only undergraduate level class in the nation that focuses on this important subject. While computer science students everywhere have received a healthy dose of Turing Machines, few have been given a chance to gain practical experience in software testing, a discipline critical to the success of their potential employers.

The driving force behind the class is lecturer Seth Ourfalian. After earning a Bachelor of Science degree in computer science at UCI, Seth spent five years developing compilers and has managed software testing groups at three high-tech companies. Seth first got the idea for the class when he heard Department Chair Dr. Michael Pazzani call for qualified people to teach practical classes at UCI. Rising to this challenge, Seth now teaches automated software testing and promotes the idea of exposing undergraduates to software testing while they are still in school. "The majority of ICS students don't know much about formal testing methodologies — and many of those who do, think it is boring or not challenging. They are unaware how important it is. Why not start them out early? That's the message. There is no reason why students should wait until they graduate to receive training."

UCI's Department of Information and Computer Science has the most undergraduate ICS majors of any campus in the University of California system, and has the third-highest number of ICS majors of any school in the West. *PC Week* recently ranked UCI's ICS department as one of the nation's top 10 campuses in preparing students for technology careers. The department has several areas of national quality: software engineering, data mining, human/computer interaction and embedded systems.

The *PC Week* report, "Making the Grade," cited the U.S. Department of Commerce's alarm at the serious shortage of skilled IT professionals, but showed hope for the future: "Leading universities are beginning to listen to CIOs and tailor curricula to produce graduates possessing the knowledge needed for today's demanding IT shops." Dr. Pazzani appreciates the recognition, and agrees with the report, "This course complements our more theoretical classes very nicely. Students learn about software life cycles, maintenance, and testability. These days, companies tend not to have formal training programs. If, in 10 weeks, a student can pick up a practical skill that illustrates a theory, I think it puts them ahead in the job market. We have been looking to offer special topics like these, and this is an excellent example."

Because Rational Visual Test features a comprehensive test programming language, it was a natural choice for ICS students learning software engineering principles.

RATIONAL
SOFTWARE

“Even if the students end up in development and not in testing, someone is going to have to test their code, and the students are now more aware of the advantages of designing testable software.”

An Extraordinary Tool for Testers

As Seth was preparing to teach the course, he considered several automated testing tools. The final decision to use Rational Visual Test was an easy one. Only Visual Test combined affordability with exceptional power and flexibility. Because Visual Test features a comprehensive test programming language, it was a natural choice for ICS students learning software engineering principles. And, as the most powerful test programming tool available, Visual Test was the only tool that fulfilled Seth's requirements for meeting the course objectives and for future student projects he was planning.

The course objectives outlined in the syllabus read like a mission statement for a corporate quality assurance department:

- Apply effective automated testing principles during the planning, development, and execution of automated test suites.
- Create application states to reduce test case development time and use the recovery system to allow unattended testing.
- Explore important testing issues such as error handling, portability, and localization.
- Write automated tests for Windows applications using the Visual Test language, Visual Test tools, and utilities.
- Deploy reusable and maintainable test suites.

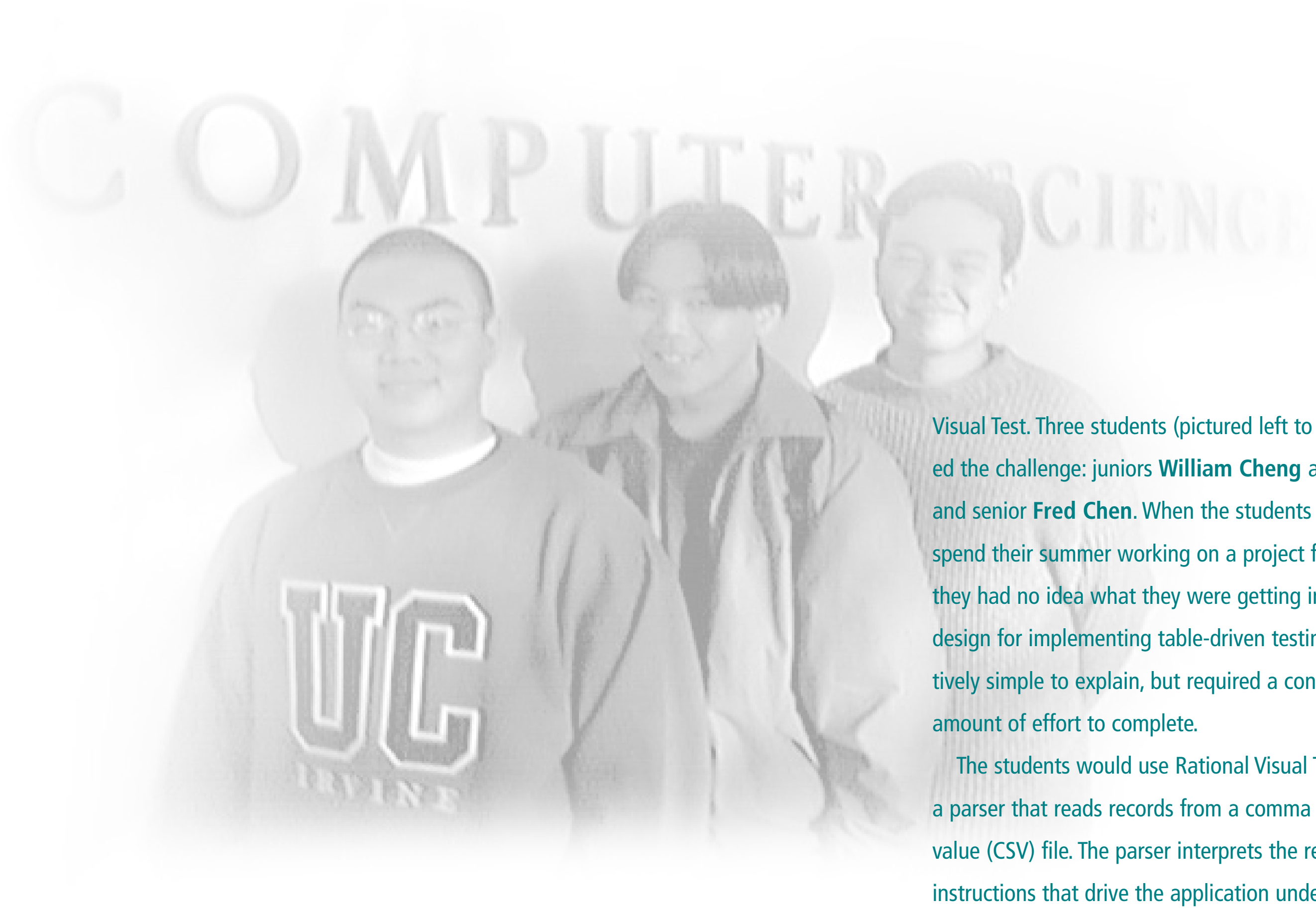
Although many of the 22 students that signed up for the four credit course had programming experience, few of them had spent much time on Windows programming. The students learned quickly though, helped by Rational Visual Test's seamless integration with Microsoft Visual Studio. One of Seth's students, Max Ho, was enthusiastic: “We found Visual Test easy to learn because it is integrated with Microsoft Visual Studio — and because the language is similar to Visual Basic it was pretty easy to pick up by going through the online help.”

Seth's industry experience led him to focus his students on writing effective test cases. Rational Visual Test's strength lies in its extensive and flexible test language. This simple language enables testing professionals to create efficient, reusable, and powerful test scripts. “If you're going to write solid test cases, they have to be well designed and programmatic.” Seth explained, “I used Rational Visual Test's Scenario Recorder as a learning tool, to record a few keystrokes and mouse clicks to see what the recorder generated.”

The course is an undeniable success. Many students listed the course on their resume to grab the attention of interviewers and recruiters. Seth understands that not every student wants to go into testing as a career, but he also believes there are substantial benefits to learning testing principles early. “Even if the students end up in development and not in testing, someone is going to have to test their code, and the students are now more aware of the advantages of designing testable software.”

Going the Extra Mile

After the spring quarter, Seth and teaching assistant Cindy Lu decided to take another step forward. Seth had attended various symposiums where speakers talked about table-driven testing in vague terms. Seth was intrigued and determined to make table-driven testing a reality with Rational Visual Test. “A live demonstration of table-driven testing is worth a thousand projected slides,” he explains. In table-driven testing, a parser program reads simple instructions from a table, typically a text file, and then executes the instructions to test an application. The table-driven methodology allows total test software reuse because all application-specific information is encapsulated in external tables. Seth invited a handful of his students to work on a 15-week independent study project on table-driven testing using Rational



Visual Test. Three students (pictured left to right) accepted the challenge: juniors **William Cheng** and **Max Ho**, and senior **Fred Chen**. When the students agreed to spend their summer working on a project for five credits, they had no idea what they were getting into. Seth's design for implementing table-driven testing was relatively simple to explain, but required a considerable amount of effort to complete.

The students would use Rational Visual Test to create a parser that reads records from a comma separated value (CSV) file. The parser interprets the records as instructions that drive the application under test and verify its functionality. The parser also handles unexpected run-time errors, restores the application under test to a default base state before and after executing each test case, and logs the test results to a text file. Table-driven testing and data-driven testing are similar concepts that differ in one important way: the data in table-driven testing incorporates decision-making logic to simulate events or to verify control properties, while the data in data-driven testing is just simple input data to the application under test.

The course is an undeniable success. Many students listed the course on their resume to grab the attention of interviewers and recruiters. Seth understands that not every student wants to go into testing as a career, but he also believes there are substantial benefits to learning testing principles early.

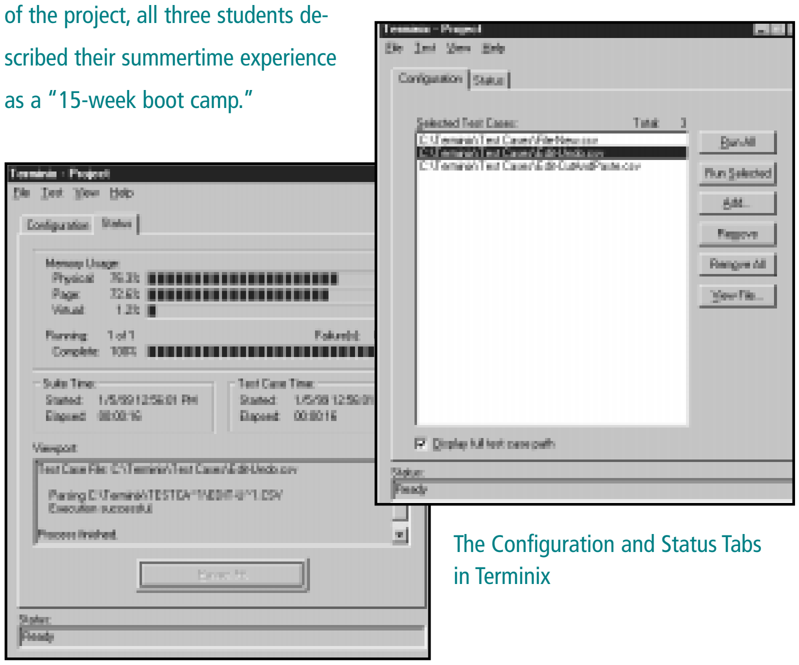
There are many advantages to the table-driven approach. First, because the individual test cases are written using a small set of simple instructions, not everyone needs to be an expert on Rational Visual Test. Test case developers can be trained quickly, and do not need to be programmers. Second, since test cases are read from text files, there is no need to recompile modified test cases. Third, the simple process of creating tables greatly accelerates the test case creation rate. And finally, the table-driven test application can be freely redistributed under the Visual Test license, which allows many testers to use it simultaneously, significantly increasing the return on investment.

As with the original class, Rational Visual Test was ideally suited to the project. Visual Test is the only testing tool that includes a GUI builder to create stand-alone GUI applications. Other more expensive tools require all tests to be run from within the tools user interface. Seth believes the simplicity of Visual Test's language gives it a big edge over tools that try to enforce object-oriented principles in testing. "Visual Test's advantage comes from a simple scripting language that is powerful at the same time. It is difficult to find experienced object-oriented programmers, let alone testers. So to use a test tool that requires object-oriented knowledge adds unnecessary complexity. Also, other tools fall short of their promise of writing test scripts once and reusing them across multiple platforms. The reality with multi-platform test environments is you write once and take your laptop." He continues, "It is easier to find people with a Visual Basic background that can jump right into it. Even if they do not have a Visual Basic background the scripting language is easy to learn, but powerful—it has pointers, callback routines and a GUI builder to create a front-end for test drivers. You can even write Windows applications with Visual Test by calling Windows APIs."

Hard Work Pays Off

The three students faced a considerable programming task as they got started. Before signing up for Seth's class, they had no experience with Rational Visual Test. What they accomplished by meeting once a week for 15 weeks is very impressive. They created "Terminix," a robust, fully functional, table-driven test driver with a GUI front-end. By the time the students completed Terminix, they had written over 13,000 lines of code and exchanged volumes of e-mail with Seth and Cindy.

For third and fourth year undergraduates in the ICS department a comment like "We've never coded so much!" does not come lightly. The students also conducted peer code reviews and spent many hours testing Terminix. Fred would integrate everyone's source code and e-mail his classmates new versions of Terminix, often during the wee hours of the morning. At the end of the project, all three students described their summertime experience as a "15-week boot camp."



Selected Test Cases: Total: 3
C:\Terminix\Test Cases\File-New.csv
C:\Terminix\Test Cases\Edit-Undo.csv
C:\Terminix\Test Cases\Edit-CutAndPaste.csv

Status: Ready

Memory Usage:
Physical 76.3%
Page 72.6%
Virtual 1.2%
Running 1 of 1 Failure(s):
Complete 100%

Suite Time:
Started: 1/5/99 12:56:01 PM
Elapsed: 00:00:16

Test Case Time:
Started: 1/5/99 12:56:01
Elapsed: 00:00:16

Viewport:
Test Case File: C:\Terminix\Test Cases\Edit-Undo.csv
Parsing C:\Terminix\TESTCA~1\EDIT-U~1.CSV
Execution successful
Process finished.

Status: Ready

The Configuration and Status Tabs in Terminix

One of the more complex components of Terminix is the parser, which reads and interprets the records in the CSV files. The format of a typical record is fairly straightforward:

Parent Window ID, Control ID, <Action>, <Parameters>

The "Parent Window ID" and "Control ID" uniquely

identify an individual Windows control, for example, a button or edit control, in the application under test. A separate mapping file is used to correlate symbolic names with application objects such as menu items, dialog boxes, and controls. “You can think of the map file as a repository of application GUI objects organized by their class,” Seth explains. The “action” field specifies an operation to perform on the control, for example, clicking the button, placing text in an edit control, or retrieving text from an edit control for verification. The “parameters” field is a string either typed into the control, or used to compare against retrieved text in order to verify it. As the project evolved, the students added more actions to the command set including a “Sleep” command to synchronize test cases with the application under test and a “Pause” command to display a dialog box for debugging purposes. The students eventually implemented “Begin Loop” and “End Loop” logic, and Max developed a mechanism by which one test case file could reference, and include, another file. This feature made it easy to write short CSV files that could drive the application under test into a particular state. These files were then called as subroutines from other test cases.

The user interface on Terminix also required a great deal of attention. The students based their design loosely on the Windows NT Task Manager program. The first tab is used for configuration, and allows the user to specify the CSV files to be executed during a test run.

“For Windows-based testing, Rational Visual Test does everything you need.”

— Seth Ourfalian, University of California, Irvine

The second tab is used to report the status of the test run, and includes vital statistics such as memory usage, percentage completion, elapsed time, and a view port that details test progress and results. Terminix also includes an Options dialog box in which the user can specify the location of the log file and toggle the use of entry and exit scripts to save and restore the state of the Windows desktop.

In addition to the parser and the user interface, the students wrote a set of event simulation wrapper functions called by the parser to drive the application under test. For the purposes of this project, the students tested a Windows text editing application. In the future, Terminix will be modified to be application independent. Terminix will also be enhanced to provide GUI coverage statistics, by reporting the total number of times each menu item, application state, or control is referenced in tables.

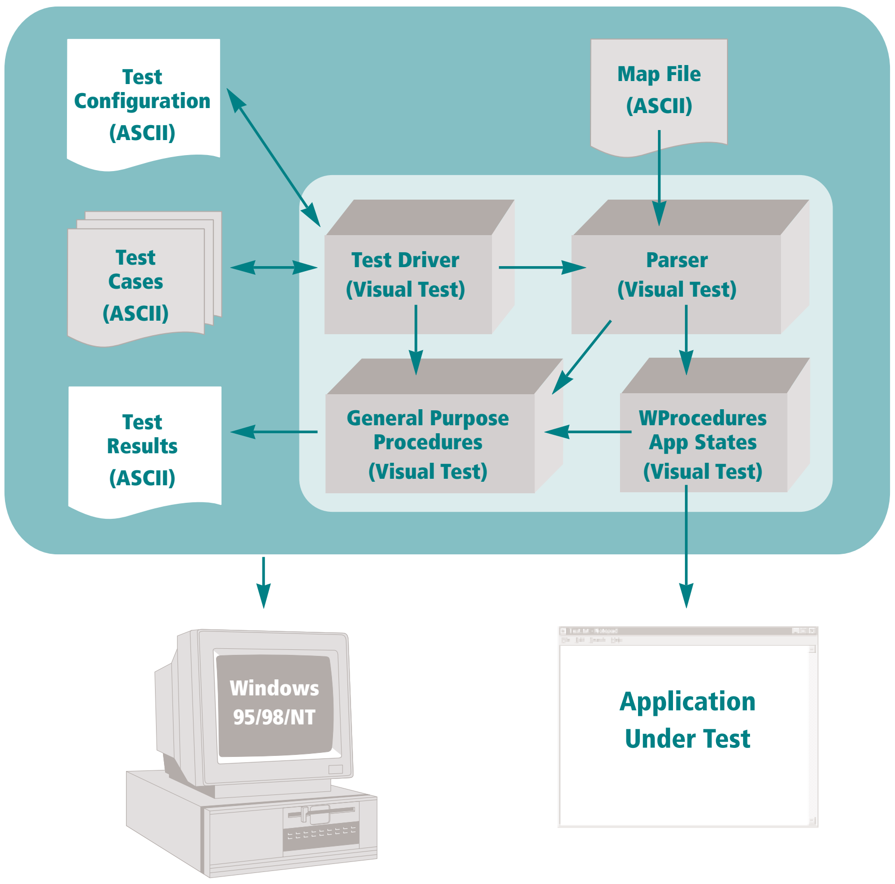

Test Configuration (ASCII)
Map File (ASCII)
Test Cases (ASCII)
Test Driver (Visual Test)
Parser (Visual Test)
Test Results (ASCII)
General Purpose Procedures (Visual Test)
WProcedures App States (Visual Test)
Windows 95/98/NT
Application Under Test

Table-Driven Automated Software Testing Architecture

The students were unanimous in their support for Rational Visual Test. They found it easy to learn, and powerful enough to do everything they needed.

Looking Ahead

Seth and Cindy are planning to make the course even more challenging in the future. For the next fall quarter, they plan to teach table-driven testing using Rational Visual Test that will include group projects for Web-based testing. They expect continued success, because everyone involved in the first course and the summer project was positive about the experience. William echoed the feelings of his classmates, "I thought it was really cool that UCI gave me the opportunity to look at software testing, because I would not have even considered the field otherwise. It was a good experience to learn Visual Test and to work as a team to produce something we think is truly useful."

In his 18 years in the industry, Seth has come across many testing tools, but only Rational Visual Test provides the programming power and simplicity to make him want to teach a course with it.

Rational Software Corporation

Corporate Headquarters
18880 Homestead Road
Cupertino, CA 95014
Toll-free: 800-728-1212
Tel: 408-863-9900
Fax: 408-863-4120
E-mail: info@rational.com
Web: www.rational.com
International Locations:
www.rational.com/corpinfo/
worldwide/locations.jtmpl