appropriate quality assurance methods are applied. For example, when a programmer writes code, one quality assurance measure is to test the code by actually running it against test data. Another would be to run special programs, called static analyzers, against the new code. A static analyzer is a program that looks for suspicious patterns in programs, patterns that might indicate an error or bug.

These two forms of quality assurance are called dynamic testing and static testing, respectively.

Software consists of discrete components or units that are eventually combined to create larger systems. The units themselves must be tested, and this process of testing individual units is called unit testing. When the units are combined, the integrated subsystems must be tested and this process of testing the integrated subsystems is called integration testing. Professor Silber told the Sentinel-Observer about his work at Silicon Techtronics: "Mike [Waterson] told me to go in there [into the company] and conduct an impartial review of his software testing procedures and to make my findings public. Mike seemed confident, perhaps because of what his managers had told him, that I would find nothing wrong with quality assurance at Silicon Techtronics."

Soon after arriving at Silicon Techtronics, Professor Silber focused his attention on procedures for dynamically testing software at the high tech company.

Assisted by a cadre of graduate students, Professor Silber discovered a discrepancy between the actual behavior of the section of program code (written by Randy Samuels) that caused the Robbie CX30 robot to kill its operator and the behavior as recorded in test documentation at Silicon Techtronics. This discovery was actually made by Sandra Henderson, a graduate student in software engineering who is completing her doctorate under Professor Silber. We interviewed Ms. Henderson in one of the graduate computer laboratories at Silicon Valley University.

"We found a problem with the unit testing," Ms. Henderson explained. "Here are the test results, given to us by Mr. Waterson at Silicon Techtronics, which are purported to be for the C [programming language] code which Randy Samuels wrote and which caused the killer robot incident. As you can see, everything is clearly documented and organized. There are two test suites: one based upon white box testing and another based upon black box testing. Based upon our own standards for testing software, these test suites are well-designed, complete and rigorous."

Black box testing involves viewing the software unit (or component) as a black box which has expected input and output behaviors. If the component demonstrates the expected behaviors for all inputs in the test suite, then it passes the test. Test suites are designed to cover all "interesting" behaviors that the unit might exhibit but without any knowledge of the structure or nature of the actual code.

White box testing involves covering all possible paths through the unit. Thus, white box testing is done with thorough knowledge of the unit's structure. In white box testing, the test suite must cause each program statement to execute at least once so that no program statement escapes execution. Sandra Henderson went on to explain the significance of software testing: "Neither black box nor white box testing 'proves' that a program is correct. However, software testers, such as those employed at Silicon Techtronics, can become quite skillful at designing test cases so as to discover new bugs in the software. The proper attitude is that a test succeeds when a bug is found."

"Basically, the tester is given a set of specifications and does his or her best to show that the code being tested does not satisfy its specifications", Ms. Henderson explained.

Ms. Henderson then showed this reporter the test results that she actually obtained when she ran the critical "killer robot" code using the company's test suites for white box and black box testing. In many cases, the outputs recorded in the company's test documents were not the same as those generated by the actual killer robot code.

During his interview with the Sentinel-Observer yesterday, Professor Silber discussed the discrepancy "You see, the software that was actually delivered with the Robbie CX30 robot was not the same as the software that was supposedly tested - at least according to these documents! We have been able to determine that the actual "killer code", as we call it, was written after the software tests were supposedly conducted. This suggests several possibilities: First, the software testing process, at least for this critical part of the software, was deliberately faked. We all know that there was enormous pressure to get this robot 'out the door' by a date certain. Another possibility is that there was some kind of version management difficulty at Silicon Techtronics, so that correct code was written, successfully tested, but the wrong code was inserted into the delivered product."

We asked Professor Silber to explain what he meant by "version management". "In a given project, a given software component might have several versions: version 1, version 2 and so forth. These reflect the evolution of that component as the project progresses. Some kind of mechanism needs to be in place to keep track of versions of software components in a project as complex as this one. Perhaps the software testers tested a correct version of the robot dynamics code, but an incorrect version was actually delivered. However, this raises the question as to what happened to the correct code."

Professor Silber sat back in his chair and sighed. "This really is a great tragedy. If the 'killer code' had gone through the testing process, in an honest manner, the robot would never have killed Bart Matthews. So, the question becomes, what was going on at Silicon Techtronics that prevented the honest testing of the critical code?"

The Silicon-Observer asked Professor Silber whether he agreed with the notion that the user interface was the ultimate culprit in this case. "I don't buy the argument, being put forth by my colleague, Professor Gritty, that all of the culpability in this case belongs to the user interface designer or designers. I agree with some of what he says, but not all of it. I have to ask myself whether Silicon Techtronics was placing too much emphasis on the user interface as a last line of defense against disaster. That is, they knew there was a problem, but they felt that the user interface could allow the operator to handle that problem. " The Silicon-Observer then asked Professor Silber about the charge made against him that he should never have accepted Waterson's appointment to conduct an impartial investigation into the accident. Critics point out that Silicon Valley University and Professor Silber, in particular, had many business ties with Silicon Techtronics, and thus he could not be counted upon to conduct an impartial investigation.

"I think my report speaks for itself," Professor Silber replied, visibly annoyed by our question. "I have told you