

Software design decisions must balance a collection of functional and non-functional software quality attributes (e.g., correctness, efficiency, testability, and analyzability) each prioritized by business needs. It is unreasonable to expect the non-functional software quality attribute of testability to dominate software design decisions in every application domain. Yet this is exactly the tacit assumption made by purveyors of general-purpose, one-size-fits-all software test development tools.

Software development organizations invest considerable resources to create domain-specific, ad hoc, software test development tools tailored specifically to meet their needs. Siddhartha is a defined, disciplined, alternative technique for developing domain-specific test development tools that fit within a software development organization's technical and business constraints.

Automated Test Driver-Oracle Synthesis

Siddhartha¹ provides essential automated test development support in the form of a program synthesizer. The program synthesizer transforms a domain-specific, formal test specification (TestSpec) into a test driver-oracle procedure (TDOP), which automates test execution and test result verification. A TDOP invokes not only the unit-under-test (UUT) but also an embedded oracle procedure (and embedded oracle data as well, depending on the application domain) that verifies whether the tested behavior of the UUT agrees with the TestSpec.

By applying the Siddhartha technique, an expert Test Engineer develops a domain-specific TestSpec TDOP synthesizer to accept and generate formal test artifacts in formats, languages, and styles already in use in her application domain. Thus, Siddhartha provides test development automation support in application domains that cannot be well-supported by general-purpose test development tools.

Figure 1 represents the most essential artifacts and processes in applying Siddhartha to an application domain. The hyperedge labeled Synthesize Driver represents the domain-specific, TestSpec TDOP synthesizer developed by the Test Engineer. All other hyperedges are assumed to be supported by processes and tools already in use in the application domain.

Siddhartha Components

Siddhartha is comprised of a domain-independent library of reusable software tool components, a reference architecture, and a process model. An expert Test Engineer follows the process model to extend the library to develop a domain-specific TestSpec TDOP synthesizer conforming to the reference architecture. The architecture of Siddhartha's library is represented in the lower portion of Figure 2. The reference architecture common to all TestSpec TDOP synthesizers is represented in the upper portion of Figure 2.

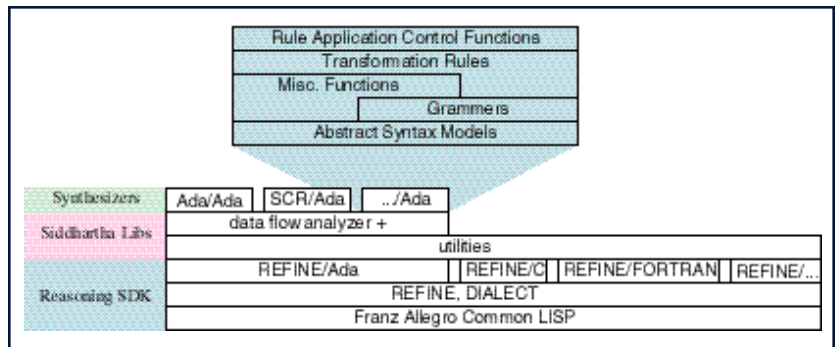


Figure 2 - Siddhartha Reference Architecture

Example Synthesizers

Siddhartha has been applied to produce two domain-specific test synthesizers to date: Siddhartha-SCR(SCR*log,Ada) and Siddhartha-regression(Ada,Ada).

Siddhartha-SCR(SCR*log, Ada)

Siddhartha-SCR(SCR*log, Ada) automatically develops requirements-based TDOPs. These TDOPs test Ada procedure UUTs against log files produced by the SCR* Toolset Simulator (available freely from U.S. Naval Research Laboratory). Most importantly, Siddhartha-SCR(SCR*log, Ada) supports Ada UUTs characterized by insufficiently defined interfaces (i.e., UUTs that rely on global variables). Siddhartha-SCR(SCR*log, Ada) uses an identifier mapping that relates variable names appearing in SCR Simulator logs to their corresponding Ada expressions in the UUT. Thus Siddhartha-SCR(SCR*log, Ada) offers both specifiers and UUT designers / implementers considerable freedom in choosing appropriate, independent data representations.

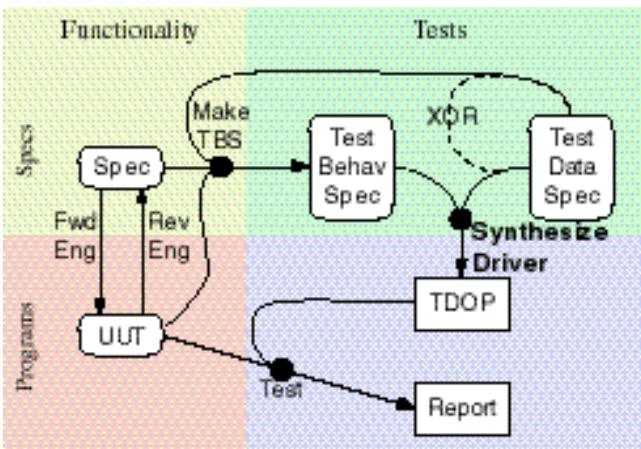


Figure 1 - Generic Siddhartha Process

Figure 3 is the instantiation of the generic Siddhartha process model to provide Siddhartha-SCR(SCR*log, Ada).

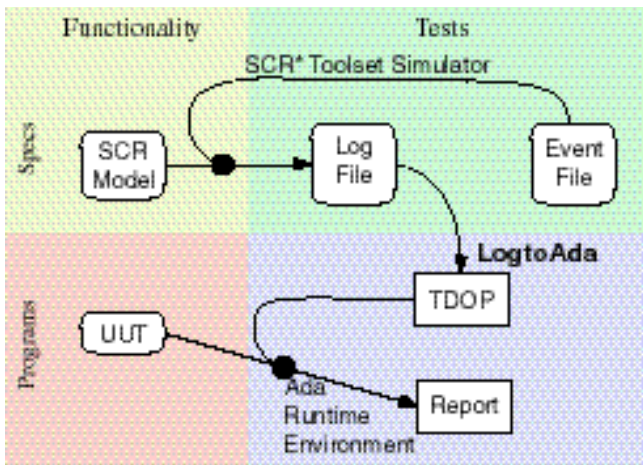


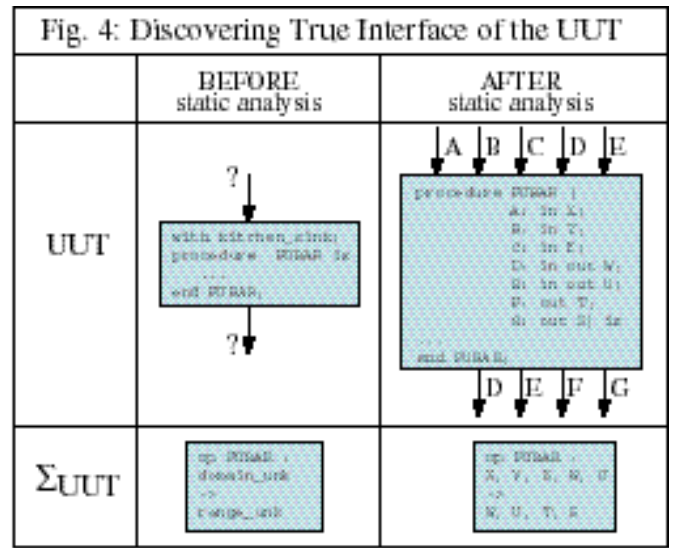
Figure 3 - Siddhartha-SCR Instantiation

Siddhartha-regression(Ada, Ada)

Siddhartha-regression(Ada, Ada) automatically develops regression-test TDOPs. These TDOPs test Ada UUTs against earlier versions, which are assumed to be functionally equivalent. Siddhartha-regression(Ada,Ada) supports UUTs with insufficiently defined interfaces (i.e., UUTs that rely on global variables). Siddhartha-regression(Ada, Ada) first uses data flow analysis to reverse engineer the effective interface of the UUT and its earlier version. This process is illustrated in Figure 4.

Siddhartha-regression(Ada, Ada) next matches the global and hidden variables and formal parameters in reverse engineered effective interfaces. Siddhartha-regression(Ada, Ada) then uses the matched, effective interface information to generate a TDOP procedure specification. The TDOP is fully encapsulated by formal

parameters. Siddhartha-regression(Ada, Ada) lastly transforms the TDOP procedure specification into a procedure body containing all the declarations and statements needed to invoke and verify UUT against its earlier version over the same matching input data.



Validated Ideas

Siddhartha has been validated against a significant, real-world example: the Ada operational flight program (OPF) for research flight control system (RFCS) of the production support flight control computer (PSFCC) used on the F/A-18B Hornet system research aircraft (SRA) operated by NASA Dryden Flight Research Center (DFRC).

Siddhartha system requirements

software

- Reasoning SDK (formerly known as Software Refinery)
- REFINE/Ada
- Solaris operating environment
- emacs or xemacs

Reasoning SDK and REFINE/Ada are available from Reasoning, Inc.

hardware

- SPARCstation
- >= 64 MB RAM
- >= 200 MB disk swap space
- 500 MB permanent disk space

'Siddhartha (si - dar' - tuh): An epithet of Buddha meaning "he who has attained." Siddhartha = SDRTA: Signature Discovery, Reformulation, Translation, & Automation.

References:

- A.A. Reyes and D.J. Richardson, Specification-Based Testing of Ada Units with Low Encapsulation, *Proceedings of the 13th IEEE International Automated Software Engineering Conference 1998*, Alex Quilici, Bashar Nuseibeh, David Redmiles, eds., Honolulu, Hawaii, October 1998.
- A.A. Reyes and D.J. Richardson, Transformational Test Driver-Oracle Synthesis, *Proceedings of the 1999 International Conference on Software Engineering-Workshop on Software Transformation Systems (ICSE-STIS 1999)*, Marcelo Sant'Anna, ed., Los Angeles, CA, U.S.A., May 1999.
- A.A. Reyes and D.J. Richardson, Siddhartha: A Technique for Building Domain-Specific Test Synthesizers, submitted to *ASE'99: 14th International Conference on Automated Software Engineering*, UCI-ICS-TechReport-99-23, Information & Computer Science, University of California, Irvine, May 1999.



The **ROSATEA** tools are research prototypes, some of which have been successfully transitioned into use on real projects, but not yet as commercial systems. These tools were developed by the Research Organization for Specification- and Architecture-based Testing E (&) Analysis at the University of California at Irvine. The work was done in conjunction with the Perpetual Testing Projects sponsored by the DARPA's EDCS program.



The **Siddhartha** research and development effort is sponsored in part by: the Air Force Materiel Command, Air Force Research Laboratory, and the Defense Advanced Research Projects Agency under agreement number #F30602-97-2-0033. The views and conclusions contained herein are those of the researchers and should not be interpreted as representing the official position or policy, either expressed or implied, of the U.S. Government, AFMC, Rome Laboratory, DARPA, or the University of California, and no official endorsement should be inferred.

Freely Available Software

Information about UC Irvine's Research Organization for Specification- and Architecture-based Testing E (&) Analysis (ROSATEA), as well as its software, is available at:
<http://www.ics.uci.edu/~rosatea/siddhartha.html>

Contact Information

Professor Debra J. Richardson
 Information and Computer Science
 University of California
 Irvine, California 92697-3425

url: <http://www.ics.uci.edu/~djr>

email: djr@ics.uci.edu
 voice: 949-824-7353
 fax: 949-824-1715