The Component Workload Emulator (CoWorkEr) Utilization Test Suite (CUTS) User's Manual

James H. Hill

January 30, 2009

Abstract

The Component Workload Emulator (CoWorkEr) Utilization Test Suite (also known as CUTS) is a model-driven engineering (MDE)-based tool that allows developers to conduct system integration test at early stages of development. Furthermore, is used to help identify system integration problems, such as failure to meet end-to-end response time due to poor infrastructure implementation, before they become to hard to locate and resolve later in the development lifecycle.

Developers and testers use CUTS to model the expected behavior of their real components and the structure of their system. The behavior models are then used to generate emulation components that are compliant with the target technologies specification and have the same interfaces and attributes as their real counterpart. The emulated components are then executed in the target, or representative, environment and different QoS metrics (e.g., throughput, response time, and service rate) are monitored and collected.

Developers and testers can grapically view the collected metrics while the system is executing to understand its current performance, and gain insight on how to improve it. Developer also have the option of viewing and analyzing collected performance metrics postmortem. Lastly, as the real component's implementation is completed, it can replace its respective emulation component to produce more realistic results and facilitate continuous system integration.

This book is the user's guide to CUTS. It contains information about building and installing CUTS, using its modeling languages to create system experiments, and details on CUTS many tools that help simpify component-based distributed system development, not only during the early stages of development, but throughout the entire development lifecycle.

Contents

I Getting Started **5**

1 Building and Installing CUTS **7**
- 1.1 CUTS Runtime Architecture . . . 7
  - 1.1.1 Downloading source files . . . 8
  - 1.1.2 Configuring your build environment . . . 8
  - 1.1.3 Building the runtime architecture . . . 8
- 1.2 CUTS Modeling Tools (Windows-only) . . . 9
  - 1.2.1 Installing prebuilt version via .msi package . . . 9
  - 1.2.2 Building from sources (advanced) . . . 9
- 1.3 CUTS Analysis Tools (Windows-only) . . . 10
  - 1.3.1 Installing prebuilt version via .msi package . . . 10
  - 1.3.2 Building from sources (advanced) . . . 10

2 Quick Start Tutorial **11**
- 2.1 Modeling Behavior and Workload . . . 11
- 2.2 Generating Test System Implementation . . . 12
- 2.3 Executing in Target Environment . . . 12
- 2.4 Analyzing Test Results . . . 12

Part I

Getting Started

Chapter 1

Building and Installing CUTS

CUTS has many small projects that comprise the entire system execution modeling (SEM) tool. Its many projects, however, can be divided into three main categories, or project groups:

- Runtime architecture
- Modeling tools
- Analysis tools

This chapter discusses how to build and install each of the aforementioned project groups for CUTS. Depending on your usage of CUTS, you may not need to build and install all projects in a category on the same machine. For example, you may install the runtime architecture of CUTS in your testing environment, and the modeling and analysis tools outside of your testing environment to minimize interference with the testing process when viewing collected performance metrics in real-time. We are aware of these needs, and have setup the build process to decouple projects within a group from projects external to its corresponding category. You can, therefore, refer to each of the following sections on building and installation in isolation since project groups do not explicitly depend on each other.

1.1 CUTS Runtime Architecture

The runtime architecture for CUTS allows developers and testers to emulate system experiments on their target architecture. CUTS also provides the mechanisms to monitor and collect performance metrics for the executing system. In order to build the CUTS runtime architecture, you will need the following technologies installed on the target machine(s):

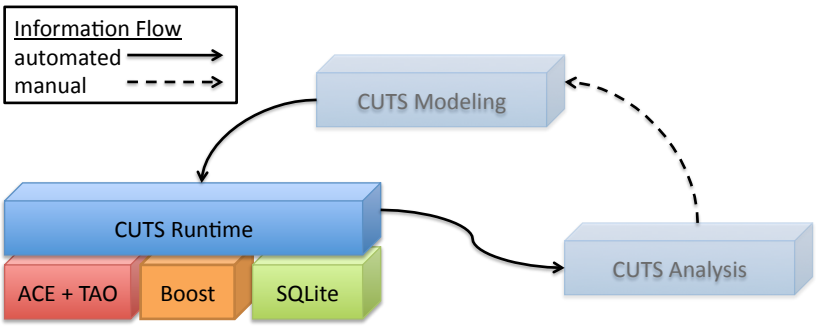

Figure 1.1. Building blocks for the CUTS runtime architecture

- **ACE + TAO** - This set of middleware is used to abstract away the complexities of implementing applications that operate on many different operating systems (ACE), and perform distributed communication (TAO). You can download ACE + TAO from the following location: `http://www.dre.vanderbilt.edu/TAO`.

- **Boost** - This set of middleware is used for implementing the parsers (Boost Spirit) used within different artifacts of CUTS. You can download Boost from the following location: `http://www.boost.org`.
- **SQLite** - This set of middleware is used to support flat file archives for test results. You can download SQLite at the following location: `http://www.sqlite.org`.
- **XSC (not pictured)** - This set of middleware is used to convert XML documents to/from objects. You can download XSC from the following location: `svn://svn.dre.vanderbilt.edu/XSC/trunk`.

1.1.1 Downloading source files

You can obtain the latest snapshot of the CUTS runtime architecture from the DOC Group Subversion repository at the following location:

`svn://svn.dre.vanderbilt.edu/DOC/CUTS/trunk/CUTS`

If you need to download a stable version of the source code, then you can access at one of the subdirectories in the repository at the following location:

`svn://svn.dre.vanderbilt.edu/DOC/CUTS/tags`

1.1.2 Configuring your build environment

CUTS inherits the configuration for ACE + TAO and XSC set during their installation, however, Boost and unixODBC (non-Windows only) remain to be configured. To configure Boost, please set the following environment variables[1]:

```
%> export PATH=$PATH:$BOOST_ROOT/lib
%> export BOOST_ROOT= location of Boost
%> export BOOST_VERSION= name of subdirectery under $BOOST_ROOT/include, e.g., boost-1_34_1
%> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$BOOST_ROOT/lib[2]
```

To configure SQLite, please set the following variables:

```
%> export SQLITE_ROOT= location of SQLite
%> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$SQLITE_ROOT/lib
```

Finally, set the environment variables for CUTS as follows:

```
%> export CUTS_ROOT= location of CUTS
%> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CUTS_ROOT/lib
%> export PATH=$PATH:$CUTS_ROOT/bin
```

1.1.3 Building the runtime architecture

We use Makefile, Workspace, Project Creator (MPC) to assist in building CUTS on different operating systems, and with different compilers. Before you can build the runtime architecture, you must first generate the target workspace. Use the following command to generate the workspace:

```
%> $(ACE_ROOT)/bin/mwc.pl -type TYPE -features boost=1,xsc=0,
sqlite3=1[,FEATURES] CUTS.mwc
```

where TYPE is your compiler type, and FEATURES are the enabled features for your build of the CUTS runtime architecture[3]. CUTS has a set of features enabled by default, which are located in `$(CUTS_ROOT)/default.features`. You can either enable features within that file, or via the command-line. Once you have generated the workspace, you can build the solution using your specified compiler.

[1] On Windows-based systems, environment variables are escaped using percent signs (`%`) and paths are seperated using semi-colons (`;`)
[2] OnWindows-based systems, please add library paths to `PATH`. On Mac OS-basedsystems, please add library paths to `DYLD_LIBRARY_PATH`.
[3] You can type `$ACE_ROOT/bin/mwc.pl --help` to view the command-line options for MPC.

1.2 CUTS Modeling Tools (Windows-only)

CUTS modeling tools provide system developers and testers with an enviroment for rapidly constructing experiments for distributed component-based systems, and generating testing system for their target architecture. The modeling tools are built on the following technologies:

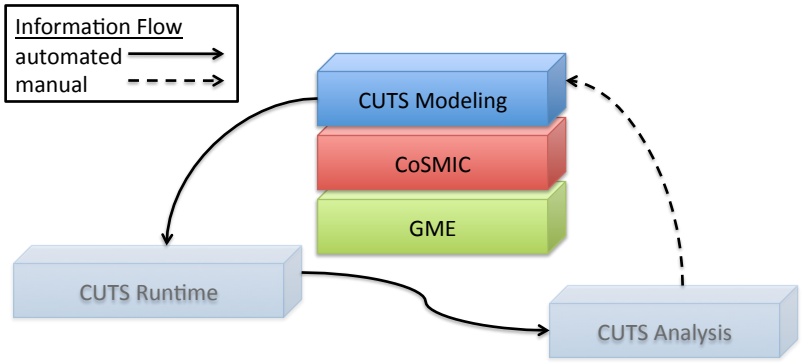

Figure 1.2. Building blocks for the CUTS modeling tools

- **GME** - The Generic Modeling Environment (GME) is a graphical modeling tool for creating domain-specific modeling languages (DSMLs). You can download GME from the following location: `www.isis.vanderbilt.edu/projects/GME`.
- **CoSMIC** - The Component Synthesis via Model Integrated Computing (CoSMIC) is a tool suite designed to address the complexities of building large-scale component-based systesm, such as sytem assembly, deployment, packaging, and planning. You can download CoSMIC from the following location: `www.dre.vanderbilt.edu/cosmic`.

1.2.1 Installing prebuilt version via .msi package

The easiest and quickest way to install the CUTS modeling tools is to install them using the .msi installer. You can download the latest version of the CUTS modeling tools from the following location: `www.dre.vanderbilt.edu/CUTS/downloads`. Before you can install the CUTS modeling tools, please make sure you have installed GME and CoSMIC. Once both GME and CoSMIC are installed (in that order), then you can install the CUTS modeling tools.

1.2.2 Building from sources (advanced)

If you choose, you can build the modeling tools from source. This is given you have downloaded all the source from the CUTS source code repository (see Section 1.1.1). To build the CUTS modeling tools, first install the latest version of GME. Then, you **MUST** build CoSMIC from source as well. This is required because the CUTS modeling tools are built on top of CoSMIC, and its therefore dependent on CoSMIC. You can find instructions for building CoSMIC from source at the following location:

```
www.dre.vanderbilt.edu/cosmic/downloads
```

After you have built and installed CoSMIC from sources, you are ready to build the CUTS modeling tools from source. Please use the following commands to build the modeling tools using your flavor of Visual C++:

```
%> cd %CUTS_ROOT%
%> mwc.pl -type [vc71 | vc8 | vc9] -features modeling=1,cosmic=1,boost=1,xsc=1,xscrt=1
CUTS_CoSMIC.mwc
%> open solution and build
```

The build process will ensure that all the interpreters are installed and configured for your environment.

1.3 CUTS Analysis Tools (Windows-only)

The CUTS analysis tools allow developers to view and analyze performance metrics of their system, and a CUTS test run. The analysis tools are built using the following key technologies:

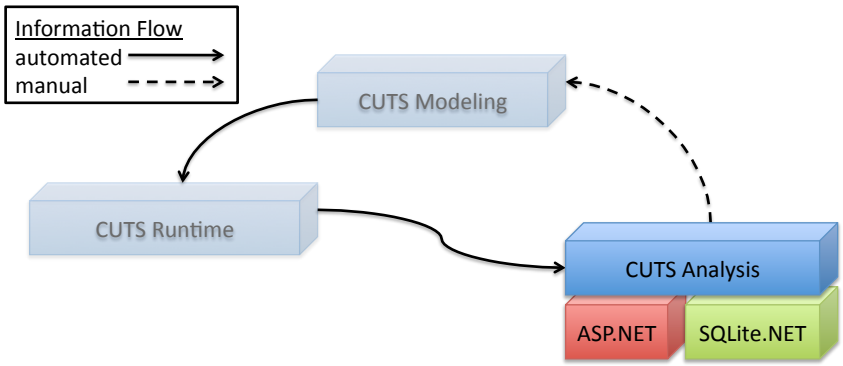

Figure 1.3. Building blocks for the CUTS analysis tools

- **ASP.NET** - ASP.NET is used to create the web application for the analysis tools, which is the main access portal for viewing test results.
- **SQLite.NET** - SQLite.NET is the embedded database technology we use to manage the backend database(s). You can download SQLite.NET from the following location: `sqlite.phxsoftware.com`

1.3.1 Installing prebuilt version via .msi package

The easiest and quickest way to install the CUTS analysis tools is to install them using the .msi installer. You can download the latest version of the CUTS analysis tools from the following location: `www.dre.vanderbilt.edu/CUTS/downloads`. Before you can install the CUTS modeling tools, please make sure you have installed ASP.NET 2.0 and SQLite.NET. Once both ASP.NET 2.0 and SQLite.NET are installed, then you can install the CUTS modeling tools [4] [5].

1.3.2 Building from sources (advanced)

If you choose, you can build the CUTS analysis tools from source. This is given you have downloaded all the source from the CUTS source code repository (see Section 1.1.1). To build the CUTS analysis tools, first install Visual Studio.NET 2003 or better. Afterwards, please use the following commands to build the analysis tools:

```
%> cd %CUTS_ROOT%\utils\BMW
%> mwc.pl -type [vc8 | vc9] BMW.mwc
%> open solution and build
```

The build process will ensure that all assemblies and the website build correctly. Once the build process is complete, you will to register the following location with Microsoft IIS: `%CUTS_ROOT%\utils\BMW\website`. This will allow you to view the website from your favorite web browser.

[4] We assume that Microsoft Internet Information Services (IIS) is already installed on the server that you installing the CUTS analysis tools. Otherwise, please install Microsoft IIS before installing the CUTS analysis tools, or any of its dependencies.

[5] We currently do not support Mono. Future versions of the CUTS analysis tools, however, will support Mono.

Chapter 2

Quick Start Tutorial

This chapter provides a quick start tutorial for using CUTS. After reading this chapter, you will have a basic understanding of how to:

1. model application behavior and workload;
2. generate a test system model the constructed model;
3. execute the system in your target environment;
4. collect and analyze test results.

In this tutorial, you will be measuring the end-to-end response time a simple client/server application that communicates using asychronous events. This tutorial assumes you have basic understanding of the Generic Modeling Environment (GME), CoSMIC, and the CORBA Component Model (CCM). More specifically, this turorial will target the Component Integrated ACE ORB (CIAO) middleware[1]. Although this tutorial targets CIAO, the experience gained from this tutorial can be applied to any architecture CUTS supports.

2.1 Modeling Behavior and Workload

Using GME, open the following file:

`$(CUTS_ROOT)/examples/PICML/GettingStarted.xme`

This model contains the structure of the client/server application. Moreover, this is usually the starting point for using CUTS within CoSMIC. Currently, application behavior and workload is modeled in the `Behavior` aspect of a component's interface definition in PICML. First, let's add behavior and workload to the client component by assuming the client component should periodically send an event to the server. As illustrated in Figure **??**, the client model already contains model elements for its input/output ports. What remains is associating the correct behavior and workload with these input/output ports for emulation purposes. Therefore, please complete the following steps:

1. Add a `PeriodicEvent` model element to the active model and set its `Period` attribute to 1000 msec.
2. Insert an `InputAction` element, changes its name to `periodicPing`, and connect the `PeriodicEvent` to the `InputAction`.
3. Insert a `State` element, connect it with the `InputAction`.
4. Insert an `OutputAction`, connect it with the `State`.
5. Insert and connect another `State`.
6. Connect the final `State` with the originating `InputAction`, *i.e.*, `periodicPing`, to signify end of the behavior.

Figure **??** illustrates a complete model for the previous steps.

[1] CIAO is a open-source implementation of the CCM and freely available for download at the following location: www.dre.vanderbilt.edu/CIAO.

2.2 Generating Test System Implementation

2.3 Executing in Target Environment

2.4 Analyzing Test Results