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

James H. Hill

September 14, 2009

Abstract

Enterprise distributed systems (such as air traffic management systems, cloud computing centers, and shipboard computing environments) are steadily increasing in size (e.g., lines of source code and number of hosts in the target environment) and complexity (e.g., application scenarios). To address challenges associated with developing next-generation enterprise distributed systems, the level of abstraction for software development steadily increases as well. Distributed system developers therefore focus more on the system's "business-logic" instead of wrestling with low-level implementation details, such as development and configuration, resource management, fault tolerance. Moreover, increase the level of abstraction for software development helps promotes reuse of the system's "business-logic" across different application domains, which inherently reduces (re)invention of core intellectual property.

Although increasing the level of abstaction for software development is improving the software lifecycle of next-generation enterprise distributed systems, system quality-of-service (QoS) properties (e.g., latency, throughput, and scalability) are not evaluated until late in the software lifecycle, i.e., during system integration time. This is due in part to the serialized-phasing developmen problem where the infrastructure- and application-level system entities, are developed during different phases of the software lifecycle. Consequently, distributed system developers do not realize the system under development does not meet its QoS requirements until its too late, i.e., during complete system integration time.

System execution modeling (SEM) tools is a model-driven engineering technique that helps overcome the effects of serialized-phasing development. SEM tools provide distributed system developers with the necessary artifacts and tools for modeling system behavior and workload, such as computational attributes, resource requirements, and network communication. The constructed models are then used to evaluate QoS properties of the system under development during early phases of the software lifecycle. This enables distributed system developers to pinpoint potential performance bottlenecks before they become to costly to locate and rectify later in the software lifecycle.

The Component Workload Emulator (CoWorkEr) Utilization Test Suite (also known as CUTS) is a SEM tool designed for next-generation enterprise distributed systems. Distributed system developers and testers use CUTS to model the expected behavior and workload of the system components under development using high-level domain-specific modeling languages. Model interpreters then transform the constructed behavior and workload models into source code for their target architecture, i.e., the components same interfaces and attributes as their real counterpart. Finally, system developers and testers emulate the auto-generated components in their target (or representative) environment and collect QoS metrics. This enables distributed system developers and testers to conduct system integration test at early stages of software lifecycle, instead of waiting until complete system integration time to perform such testing.

This book therefore is the end-user's guide to CUTS. It contains information on building and installing CUTS, using its domain-specific modeling languages to model behavior and workload, creating system experiments, and details on CUTS many tools that help simpify evaluting QoS of next-generation distributed systems continuously throughout the software lifecycle.

Contents

I Getting Started 5

1 Building and Installing CUTS 7
- 1.1 CUTS Runtime Architecture . . . 7
  - 1.1.1 Obtaining Source Files . . . 8
  - 1.1.2 System Configuration . . . 8
  - 1.1.3 Installation . . . 8
- 1.2 CUTS Modeling Tools (Windows-only) . . . 8
  - 1.2.1 Installing Prebuilt Version . . . 9
  - 1.2.2 Building from Sources (advanced) . . . 9
- 1.3 CUTS Analysis Tools (Windows-only) . . . 9
  - 1.3.1 Installing Prebuilt Version . . . 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 . . . 13

II Appendix 15

A Building and Installing Third-Party Libraries 17
- A.1 Boost . . . 17
  - A.1.1 Installation . . . 17
  - A.1.2 System Configuration . . . 17
- A.2 Xerces-C . . . 18
  - A.2.1 Installation . . . 18
  - A.2.2 System Configuration . . . 18
- A.3 ACE, TAO, CIAO, and DAnCE . . . 18
  - A.3.1 System Configuration . . . 18
  - A.3.2 Installation . . . 19
- A.4 XML Schema Compiler (XSC) . . . 20
  - A.4.1 System Configuration . . . 20
  - A.4.2 Installation . . . 20
- A.5 Perl-Compatible Regular Expressions (PCRE) . . . 20
  - A.5.1 Installation . . . 20
  - A.5.2 System Configuration . . . 21
- A.6 SQLite . . . 21
  - A.6.1 Installation . . . 21
  - A.6.2 System Configuration . . . 21

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):

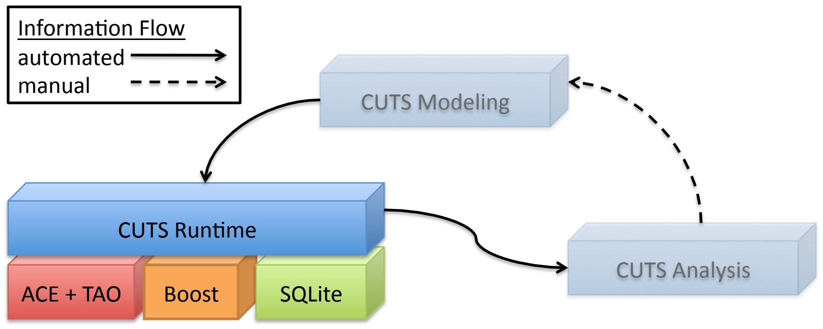

Figure 1.1. Building blocks for the CUTS runtime architecture

- **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 libraries 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 libraries is used to support flat file archives for test results. You can download SQLite at the following location: `http://www.sqlite.org`.
- **PCRE (not pictured)** - This set of libraries is used to enable PERL regular expression support in CUTS. You can download PCRE at the following location: `http://www.pcre.org`.
- **XSC (not pictured)** - This set of libraries and applications 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 Obtaining 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 System Configuration

Before you can build CUTS, you must for configure you environment. Please set the following environment variables:

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

Please see Appendix A for instructions for configuring, building, and installing third-party libraries needed by CUTS.

1.1.3 Installation

We use Makefile, Workspace, Project Creator (MPC) [1] 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 FEATURES] CUTS.mwc
```

where TYPE is your compiler type, and FEATURES is a comma-separated list of enabled/disabled features for your build of the CUTS runtime architecture.[2] 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.

1.2 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:

[1] For more information on MPC, please see the following location: `http://www.ociweb.com/products/mpc`

[2] You can type `$ACE_ROOT/bin/mwc.pl --help` to view the command-line options for MPC.

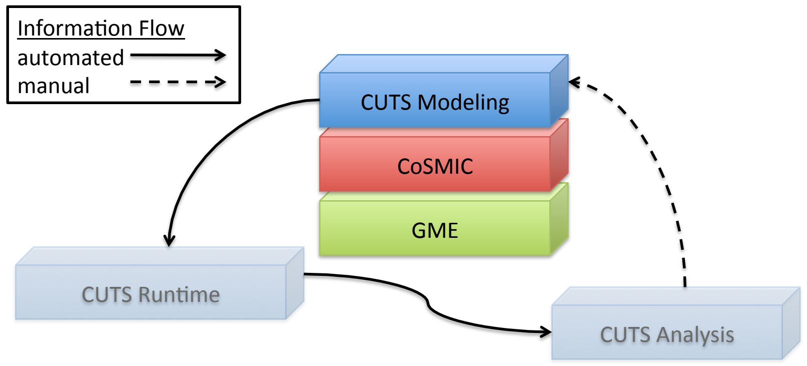

Figure 1.2. Building blocks for the CUTS modeling tools

- **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: `http://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: `http://www.dre.vanderbilt.edu/cosmic`.

1.2.1 Installing Prebuilt Version

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: `http://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:

`http://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:

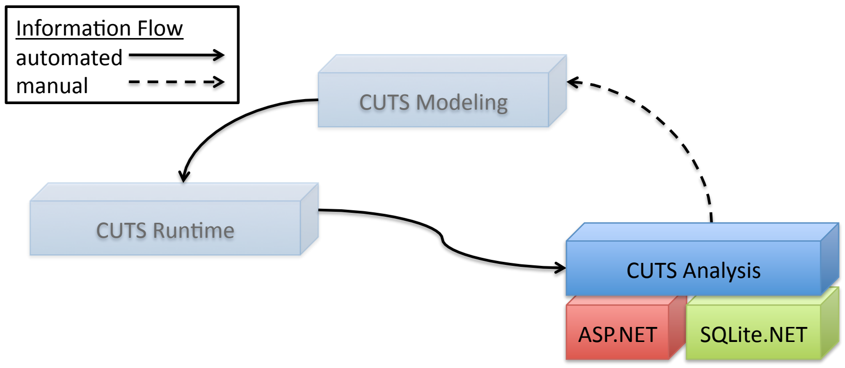

Figure 1.3. Building blocks for the CUTS analysis tools

- **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: `http://sqlite.phxsoftware.com`

1.3.1 Installing Prebuilt Version

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:

`http://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. [3] [4]

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.

[3] 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.

[4] 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/PICML, 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, and is usually the starting point for using CUTS within PICML. Currently, application behavior and workload is modeled in the `Behavior` aspect of a component's interface definition in PICML. The components in the model located at:

`GettingStarted/InterfaceDefinitions/GettingStarted/GettingStarted`

already contain model elements for its input/output ports. What remains is associating the correct behavior and workload with these input/output ports for emulation purposes. First, let's add behavior and workload to the client component by assuming the client component should periodically send an event to the server. Please complete the following steps:

1. Add a `PeriodicEvent` model element to the active model and set its name to `periodicPing` and its `Hertz` attribute to 10 (*i.e.*, 10 events/sec).
2. Insert an `InputAction` element, change its name to `periodicPing`, and connect the `PeriodicEvent` to the `InputAction`.
3. Insert a `State` element, connect it with the `InputAction`.

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

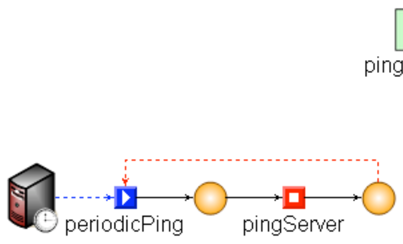

Figure 2.1. Client behavior model in PICML for CUTS emulation.

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 2.1 illustrates the complete behavior model of the client in PICML, which is implemented in the previous steps.

2.2 Generating Test System Implementation

After modeling the behavior and workload, the next step is to generate source code from the model. This will enable emulation of the test system on its target architecture. To generate source code from the model, launch the CUTS interpreter and excute the following steps (also illustrated in Figure **??**):

1. Select `Generate component implementation` radio button;
2. Enter the target output directory;
3. Select `Component Integrated ACE ORB (CIAO)` in the listbox;
4. Click the `OK` button.

Once the CUTS interpreter finishes generating source code, the IDL and CIDL files need to be generated in order to compile the source code. These files are created by the IDL Generator and CIDL Generator interpreters, respectively, provided with CoSMIC [2]. Finally, there will be a `GettingStarted.mwc` file (its name may be different) located in the output directory selected for source code generation. This is a Makefile, Project and Workspace Creator (MPC) workspace file that contains all the necessary information to sucessfully compile the generated source code. Use `ettingStarted.mwc` to generate the appropriate workspace and then compile it.

2.3 Executing in Target Environment

One design goal of CUTS is to (re)use the same infrastruture used in the target environment. The `GettingStarted.xme` example uses the Deployment And Configuration Engine (DAnCE), which is included with CIAO's standard distribution, to deploy the test system. To deploy the CUTS stock quoter example, use the following steps [3]:

1. Generate the flat deployment plan descriptors from the `GettingStarted` model. You can find instructions on generating the flat deployment plan in the original stock quoter tutorial included with CoSMIC.
2. Use DAnCE to deploy the GettingStarted.cdp. The original stock quoter tutorial in CoSMIC has instructions on how to deploy a solution. Be sure, however, to update the `nodemap.dat` file to the appropriate hosts and their location.

When the CUTS version of the GettingStarted example is deployed, the test manager will periodic collect log messages from each host/client, and insert them into a SQLite database for postprocessing.

[2]Please ensure to generate the IDL and CIDL files in the same directory as the source code previously generated from the model.

[3]CUTS has a distributed testing framework that simplifies many of the complexities associated with running distributed system tests. We will cover the CUTS testing framework in later chapters

2.4 Analyzing Test Results

Part II

Appendix

Appendix A

Building and Installing Third-Party Libraries

The following appendix contains best practices for building and installing third-party libraries used by CUTS projects. It is not a requirement to install third-party libraries per the instructions in this appendix. If you are experiencing compilation problems with CUTS due to third-party libraries, however, the instructions in this appendix may be of assistance. To make help explain the installation process of the thirdparty libraries, it will be assumed that all third-party libraries will be installed at the following location: `/opt/thirdparty`.[1]

A.1 Boost

The Boost libraries are required when building and installing CUTS. All artifacts for Boost, such as prebuilt versions and source files, can be downloaded from the following location: `http://www.boost.org`.

A.1.1 Installation

Non-Windows

If you are installing Boost on a non-Windows system, *e.g.*, Linux, Soloris, or MacOS X, then you can download the source code directly from its main website (`http://www.boost.org`), build it, and install it using standard procedures.

Windows

Installing Boost from sources on Windows is not a trivial process. Moreover, the prebuilt version of Boost available for download from the main website does not contain the correct configuration or directory layout. It is therefore recommended that you download and use the prebuilt version of Boost available for download from the CUTS website at the following location: `http://www.dre.vanderbilt.edu/CUTS/downloads/thirdparty/boost`. Please download the correct version of Boost that matches your version of Microsoft Visual C++ and extract it to a common location on disk.

A.1.2 System Configuration

To build CUTS correctly, the Boost environment variables must be defined. Please define the following required environment variables:

`%> 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:$BOOST_ROOT/lib`[2]
`%> export PATH=$PATH:$BOOST_ROOT/lib`

[1] This location is representative of a common location for installing third-party libraries, and may be different on your system.

[2] OnWindows-based systems, please add library paths to `PATH`. On Mac OS-basedsystems, please add library paths to `DYLD_LIBRARY_PATH`.

In some cases, Boost libraries will have an optional configuration appended to their filename, such as `-mt-d`. To use a specific configuration of Boost, the `BOOST_CFG` environment variable must be defined. The following is an example of defining the Boost configuration environment variable:

```
%> export BOOST_CFG=-mt-d
```

The configuration of Boost is now complete. Once all the other third-party libraries are installed you will be able to build CUTS (see Chapter 1).

A.2 Xerces-C

Xerces-C 3.x is required when building and installing CUTS and other third-party libraries. All artifacts for Xerces-C, such as prebuilt versions and source files, can be downloaded from the following location: `http://xerces.apache.org/xerces-c`.

A.2.1 Installation

Non-Windows

If you are installing Xerces-C on a non-Windows system, *e.g.*, Linux, Soloris, or MacOS X, then you can download the source code directly from its main website (`http://xerces.apache.org/xerces-c`), build it, and install it using standard procedures.

Windows

Installing Xerces-C from sources on Windows is not a trivial process. Moreover, the prebuilt version of Xerces-C available for download from the main website does not contain the correct configuration or directory layout. It is therefore recommended that you download and use the prebuilt version of Xerces-C available for download from the CUTS website at the following location: `http://www.dre.vanderbilt.edu/CUTS/downloads/thirdparty/xerces-c`. Please download the correct version of Xerces-C that matches your version of Microsoft Visual C++ and extract it to a common location on disk.

A.2.2 System Configuration

To build CUTS correctly, the Xerces-C environment variables must be defined. Please define the following required environment variables:

```
%> export XERCESCROOT= location of Xerces-C
%> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$XERCESCROOT/lib
```

The configuration of Xerces-C is now complete. Once all the other third-party libraries are installed you will be able to build CUTS (see Chapter 1).

A.3 ACE, TAO, CIAO, and DAnCE

ACE and TAO are required when building and installing CUTS. All artifacts for ACE and TAO can be downloaded from the following location: `http://www.dre.vanderbilt.edu/TAO`. Although CUTS is built on ACE and TAO, you can install CIAO and DAnCE as well. The following section therefore covers building and installing ACE and TAO, as well as, CIAO and DAnCE.

A.3.1 System Configuration

Unlike Boost, you must configure you system installing ACE and TAO. Once you have downloaded the ACE and TAO tarball from the website above [3], please define the following environment variables:

[3]It is recommended you download either the latest snapshot or beta version of ACE and TAO tarball. This will ensure you have the must up-to-date version of ACE and TAO that is compatible with CUTS.

```
%> export ACE_ROOT= location of ACE
%> export TAO_ROOT= location of TAO
%> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ACE_ROOT/lib
%> export PATH=$PATH:$ACE_ROOT/lib
```

After defining the ACE and TAO environment variables, the next step is configuring the middleware for the target platform. This is accomplished by defining the `$ACE_ROOT/ace/config.h` header file, which will include the correct configuration for your platform. The included configuration file has the form `config-*`, where ⋆ is the platform of choice. The following is an example `config.h` file:

```
// -*- C++ -*-

#ifndef _ACE_CONFIG_H_
#define _ACE_CONFIG_H_

#include ‘‘config-linux.h’’

#endif // !defined _ACE_CONFIG_H_
```

Non-Windows

Configuring ACE and TAO on non-Windows systems requires one more step. The configuration for the build engine also needs to be defined. Please use the following step to define the build engine's configuration:

```
%> cd $ACE_ROOT/include/makeinclude
%> ln -s platform_*.GNU platform_macros.GNU
```

where ⋆ is the architecture for your platform of choice.

CIAO and DAnCE (optional)

If you choose to install CIAO and DAnCE, then please download the complete source from the following location: http://www.dre.vanderbilt.edu/CIAO. After you have download the source files, then define the following environment variables in addition to the ones above:

```
%> export CIAO_ROOT= location of CIAO
%> export DANCE_ROOT=$CIAO_ROOT/DAnCE
%> export PATH=$PATH:$CIAO_ROOT/bin:$DANCE_ROOT/bin
```

A.3.2 Installation

After you have configured your system for ACE and TAO, you are ready to build and install ACE and TAO. Please use the following steps to build and install ACE and TAO:

```
%> cd $TAO_ROOT
%> $ACE_ROOT/bin/mwc.pl -type [build tool] TAO_ACE.mwc
```

This will generate the workspace for building and installing ACE and TAO. The name of the workspace depends on the build tool specified when generating the workspace. Finally, using your build tool of choice, you can build and install ACE and TAO.

CIAO and DAnCE (optional)

If you are installing CIAO and DAnCE, then you can use the following steps instead to build ACE, TAO, CIAO and DAnCE. First, you must build the CIDL compiler using the following steps:

```
%> cd $CIAO_ROOT/CIDLC
%> $ACE_ROOT/bin/mwc.pl -type [build tool]
 -features boost=1,exceptions=1,cidl=1 CIDLC.mwc
```

This will generate the workspace for building and installing CIDL compiler. The name of the workspace depends on the build tool specified when generating the workspace for the CIDL compiler. After you have built the CIDL compiler, you can build ACE, TAO, CIAO and DAnCE using the following steps:

```
%> cd $CIAO_ROOT
%> $ACE_ROOT/bin/mwc.pl -type [build tool] CIAO_TAO_DAnCE.mwc
```

This will generate the workspace for building and installing ACE, TAO, CIAO, and DAnCE. The name of the workspace depends on the build tool specified when generating the workspace. Finally, using your build tool of choice, you can build and install ACE, TAO, CIAO, and DAnCE.

A.4 XML Schema Compiler (XSC)

XSC is required when building and installing CUTS. All artifacts for XSC can be downloaded from the following location: svn://svn.dre.vanderbilt.edu/XSC/trunk. The remainder of this section discusses how to build and install XSC.

A.4.1 System Configuration

You must configure you system installing XSC. Once you have downloaded XSC from the website above please define the following environment variables:

```
%> svn co svn://svn.dre.vanderbilt.edu/XSC/trunk XSC
%> export XSC_ROOT= location of XSC
%> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$XSC_ROOT/lib
%> export PATH=$PATH:$XSC_ROOT/bin
```

A.4.2 Installation

After you have configured your system for XSC, you are ready to build and install it. Please use the following steps to build and install XSC:

```
%> cd $XSC_ROOT
%> $ACE_ROOT/bin/mwc.pl -type [build tool] -features xsc=1,xerces3=1 XSC.mwc
```

This will generate the workspace for building and installing XSC. The name of the workspace depends on the build tool specified when generating the workspace. Finally, using your build tool of choice, you can build and install XSC.

A.5 Perl-Compatible Regular Expressions (PCRE)

PCRE is required when building and installing several projects in CUTS. You can download the source code for PCRE from the following location: http://www.pcre.org

A.5.1 Installation

Non-Windows

If you are installing PCRE on a non-Windows system, *e.g.*, Linux, Soloris, or MacOS X, then you can download the source code directly from its main website (http://www.pcre.org), build it, and install it using standard procedures.

Windows

Installing PCRE from sources on Windows is not a trivial process. It is therefore recommended that you download and use the prebuilt version of PCRE available for download from the CUTS website at the following location: `http://www.dre.vanderbilt.edu/CUTS/downloads/thirdparty/pcre`. Please download the correct version of PCRE that matches your version of Microsoft Visual C++ and extract it to a common location on disk.

A.5.2 System Configuration

To build CUTS correctly, the PCRE environment variables must be defined. Please define the following required environment variables:

```
%> export PCRE_ROOT= location of PCRE
%> export PCRE_VERSION= version of PCRE, blank if not applicable
%> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PCRE_ROOT/lib
```

The configuration of PCRE is now complete. Once all the other third-party libraries are installed you will be able to build CUTS (see Chapter 1).

A.6 SQLite

SQLite is required when building and installing several projects in CUTS. You can download the source code for SQLite from the following location: `http://www.sqlite.org`

A.6.1 Installation

Non-Windows

If you are installing SQLite on a non-Windows system, *e.g.*, Linux, Soloris, or MacOS X, then you can download the source code directly from its main website (`http://www.sqlite.org`), build it, and install it using standard procedures.

Windows

Installing SQLite from sources on Windows is not a trivial process. It is therefore recommended that you download and use the prebuilt version of SQLite available for download from the CUTS website at the following location: `http://www.dre.vanderbilt.edu/CUTS/downloads/thirdparty/sqlite`. Please download the correct version of SQLite that matches your version of Microsoft Visual C++ and extract it to a common location on disk.

A.6.2 System Configuration

To build CUTS correctly, the SQLite environment variables must be defined. Please define the following required environment variables:

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

The configuration of SQLite is now complete. Once all the other third-party libraries are installed you will be able to build CUTS (see Chapter 1).