Weevil - User Manual

Giovanni Toffetti Carughi

January 28, 2008

Contents

1 Weevil Overview

Weevil simplifies running repeatable experiments on distributed system by modelling the system under evaluation (SUE), the test environment (testbed), and the experiment workload. It is composed of two executables:

1. *weevilgen* uses a simulator to automatically generate workloads given a simple implementation of the system clients (actors).
2. *weevil*, given a configuration file representing the experiment model, will :
   (a) create start and stop scripts for each component
   (b) deploy the experiment binaries across the testbed
   (c) start experiment components and actors
   (d) collect component logs
   (e) clean the testbed

2 Technical info

Provider USI

Introduction Software to simplify the execution and collection of results of experiments on distributed testbeds.

Development status The complete software is available for download, current Version is 1.2.2

Intended audience Developers who intend to test a distributed system on a real environment

License OSS, both GPL 2.0 and 3.0 apply.

Language M4, C++, Shell Script, Make

Sw operating language Shell, SSH, (Java)

Environment (set-up) The following components need to be installed in order to compile and run Weevil:

- SSim discrete-event library[1]

Platform N/A

Download Weevil is available from the WP4 tools repository web site at URL:
`http://plastic.isti.cnr.it/download/tools`

Documents see WP4 tools repository web site and deliverables D4.1 and D4.2

Tasks N/A

Bugs N/A

Patches N/A

Contact `toffettg@lu.unisi.ch`

3 Deployment

3.1 Install

Weevil is distributed as source code, as such the typical "configure" and "make" procedure has to be executed for installation.

The simplest way to compile the package is:

1. 'cd' to the directory containing the package's source code, cd to the 'build' directory and type '../configure' to configure the package for your system.

   The execution of weevil needs the SSim discrete-event library. So you need to have the library installed somewhere, make sure to pass the right path to the –with-ssim= option in the 'configure' command. Here is an example:

   ../configure –with-ssim=/usr/lib/ssim-1.5.0

2. Type 'make' to compile the package.

3. Type 'make install' to install.

Additional configuration information is contained in the "readme" file distributed with the source code.

3.2 Configure

Weevil directly supports experiment deployment and execution. The overall process is depicted in Figure 1. Actions are represented by rectangles and are labeled by circled numbers. Input and output data for those actions are represented by ovals. Dark ovals represent input models provided by the engineer. White ovals represent control scripts and data files generated by Weevil. The cross-hatched ovals represent data generated by the SUE during an experiment. Solid arrows represent normal input/output data flow, whereas dotted arrows represent the execution of scripts.

The following steps need to be taken to setup and conduct an experiment.

[1] available from http://www.inf.unisi.ch/faculty/carzaniga/ssim/index.html

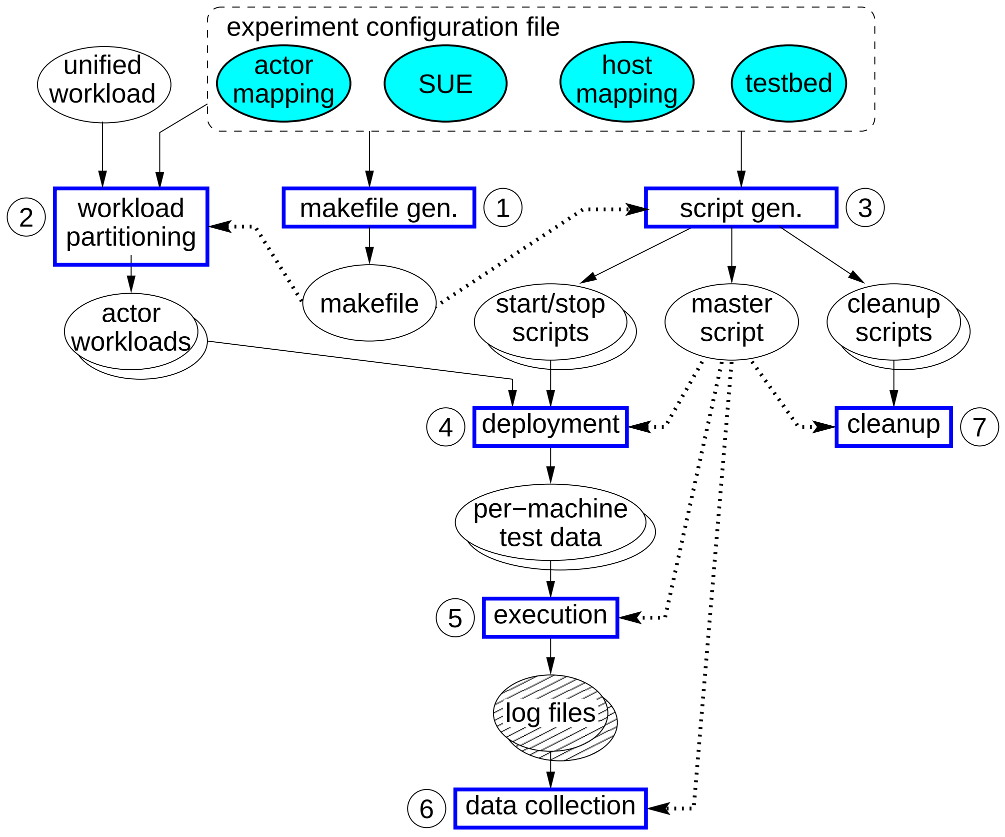

Figure 1: Weevil Experimentation Process

3.2.1 Weevil Configuration (Experiment Registration)

Create a name for the experiment (refered to as <*experiment name*> below), like "`experiment`" in the siena example; add the name into "*weevil.conf*" file after "`weevil_EXPERIMENTS:=`". You could have more than one experiment registered in "*weevil.conf*" separated with space. To start a new line, \ is needed to end the current line. Weevil only conducts experiments that have been registered in "*weevil.conf*".

For example, in file "*weevil.conf*" of the Siena example we defined one experiment:

```
weevil_EXPERIMENTS  := experiment
```

For each experiment, Weevil must be provided with a workload file (refer to Section 3.2.2) and an experiment configuration file named "<*experiment name*>*.m4*" including all the experiment configurations as illustrated in Section 3.2.3.

3.2.2 Workload File

You can generate the workload file with *weevilgen*. Of course, you can always use workloads from other workload generators or just use a real trace as the workload. But please note that the workload should be made up of workload lines with the following format:

```
event(<time stamp>, <workload process ID>, '<event content>')
```

3.2.3 Experiment Configurations

These configurations are represented by the dark ovals along the top of Figure 1. They are programmed in GNU m4 by parameterizing Weevil-defined declaration macros (Please refer to the "Weevil's Experiment

Environment Declaration Macros"[2]) to instantiate elements of two conceptual models (SUE and testbed) and necessary mappings (Please refer to the "Weevil's Experiment Environment Conceptual Models"[3]). In other words, these declaration macros will define a set of macros serving as properties of an experiment. (Please refer to the "Weevil's Experiment Environment Declaration Macros" and the "Some Other Weevil-Defined Property Macros"[4]) for these property macros.) The order of the declarations does not matter. A macro can be used before it is defined as long as it is defined somewhere in the experiment configuration file. Weevil supports the engineer during this activity by performing extensive checks on the syntax and consistency of the configurations and by providing detailed error messages about any problems it encounters. Please refer to the user manual for a complete description of the available experiment configuration macros.

4 Tutorial

As a tutorial, we use the "Siena" example distributed with the software package, which is also the introductory example in the Weevil manual. Please refer to the manual for complete reference. The aim of this simple example is to introduce the concepts and models needed to specify an experiment run. In the test, 3 instances of a Siena server (a distributed publish-subscribe system) are started and three small Java applications (*Actors*) are used to impersonate clients subscribing predicates and publishing notifications.

4.1 Workload

Two automatically generated workload files are included:

- MyWorkload.wkld
- wkld_sequential.wkld

The file MyWorkload.wkld consists of a list of events to be executed by the actor components, a brief extract from it is shown in Figure 2.

```
event(100,C1,`SUB(0, `"i tedious < 91"')',,)
event(102,C3,`SUB(0, `"i tedious > 55"')',,)
event(103,C2,`PUB(0, `"i hedge 95 & i tedious 63"')',,)
```

Figure 2: An example of workload

Each event is composed of a relative execution time w.r.t. the experiment start, an identifier of the actor that will enact the event, and an application specific command that has to be interpreted by the actor. For instance, in our example of Figure 2, the first line states that at time T=100, actor C1 is to subscribe with predicate "tedious <91".

Both workload files can be used as is for experiment runs, and provide an example of how to specify a generic workload. They have been automatially generated from MyWorkload.m4 and wkld_sequential.m4 using the command "weevilgen". You might want to postpone the study of workload generation and skip to Section 4.2 to concentrate first on actual experiment runs.

4.1.1 Simulation-Based Workload Generation

Weevil's simulation-based workload generation process supported by the package *weevilgen* is illustrated in Figure 3.

In detail, the following steps need to be taken.

[2]see doc/weevilexec/defs.html
[3]see doc/weevilexec/model.html
[4]doc/weevilexec/macros.html

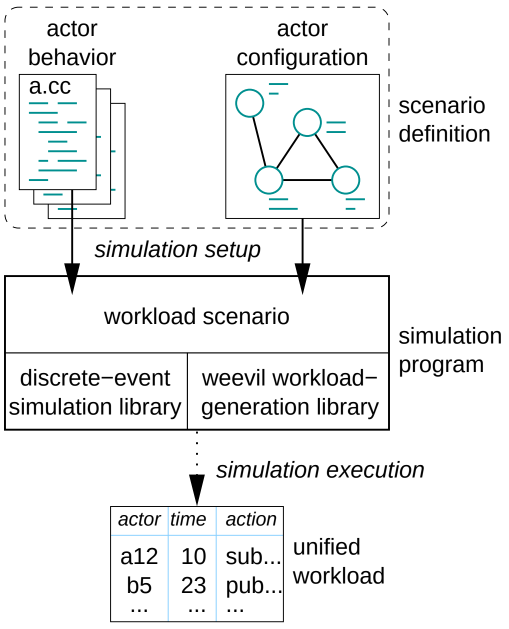

Figure 3: Simulation-Based Workload Generation

Weevilgen Configuration (Workload Registration) Create a name for the workload (refered to as <*workload name*> below), like "`MyWorkload`" in the Siena example; add the name into "*weevil.conf*" file after "`weevilgen_WORKLOADS:=`". You could have more than one workload registered in "*weevil.conf*" separated with space, as indicated in the apachesquid example. To start a new line, \ is needed to end the current line. Weevilgen only generates workloads that have been registered in "*weevil.conf*".

Then, to generate each workload, Weevil must be provided with programmed actor behavior models and an actor configuration file named "<*workload name*>*.m4*" including all the actor configurations.

Actor Behavior Model Programming One or more types of actor behaviors could be programmed in C++ with the support of Weevil's workload-generation library and SSim simulation library, and may therefore execute arbitrary functions and maintain arbitrary state. SSim is a discrete-event simulation library that supports message communication between simulated processes, so actor behavior programs may specify interactions with other actors. Weevil's workload-generation library extends the SSim's library by providing a workload-output method and convenient methods for dealing with processes by ids. Please refer to the *Weevil's workload-generation library* and the *SSim's simulation library* in the reference manual when programming your actor behavior model.

Actor Configurations After programming the actor behavior models, you can populate a scenario consisting of many actor instances specified in the actor configuration file named "<*workload name*>*.m4*". The actor behavior models and the actor configurations make up a workload scenario definition.

Figure 4 shows the portion of the Weevil conceptual model concerning the definition of workload scenarios. You need to parameterize a set of GNU m4 declaration macros in the actor configuration file. The order of the declarations does not matter. A macro can be used before it is defined as long as it is defined somewhere in the actor configuration file. Weevil supports the engineer during this activity by performing extensive checks on the syntax and consistency of the configurations and by providing detailed error messages about any problems it encounters.

Workload generation When a workload has been registered in the *weevil.conf* file, actor behaviours have been implemented, and actor configurations have been provided, the command *weevilgen* can be used to transform the actors configuration file into a complete workload.

For instance, in the "Siena" example directory type: "weevilgen gen-MyWorkload".

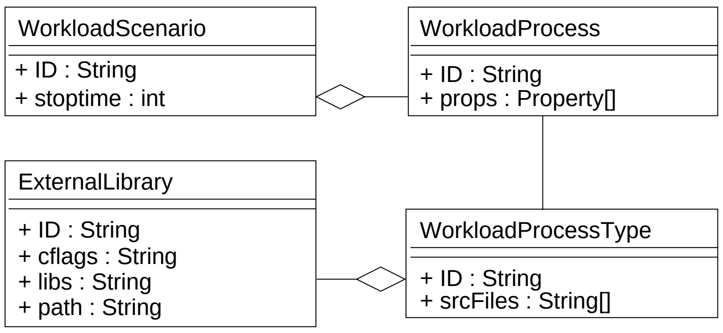

Figure 4: Workload Scenario Conceptual Model

4.2 Experiment

The configuration file for the the actual experiment run is “experiment.m4”. It specifies the different features of an experiment such as: the testbed, the workload, the number and software for components and actors. Once more, we warmly recommend you read the Weevil manual (or the related conference papers) to get confident with the concepts and terminology.

The experiment starts 3 instances of a Siena server and uses three actors to impersonate the workload and provide stimuli to the servers.

Before running the experiment you should configure it to your particular setup. In particular, you should change the *WVL_SYS_Host* definitions to match hosts you can access via ssh.

Also we recommend you setup the testbed so that you can use your public key to ssh to remote hosts. Run ssh-add to open your keystore before launching the experiment run

To run the experiment:

1. ”cd” to the experiment directory
2. type ”weevil setup-experiment”
3. If the experiment is set up correctly type ”weevil run-experiment”

While the experiment is running you can follow the experiment evolution in a different console with the command:

```
tail -f experiment.runlog
```

The typical output of the Siena experiment is reported below.

```
deploying experiment files...
The deploy copy time is X seconds
starting S0...
starting S1...
starting S2...
monitoring hosts...
communicating with each host on the testbed to
      estimate the test start time...
The deploy time is 10 seconds
The test will start at 11/30/2007 16:4:59
The deploy and execution time is 23 seconds
killing processes...
stopping system components...
copying logs back...
```

Server logs and error output will be copied to the local directory, actors output is in weevil-parallel/actorstartoutput/*/ActorID.out