cew: A C++ Component Exerciser Workbench

Peter Walsh and Jim Uhl

Department of Computing Science

Malaspina University College,

Nanaimo, B.C., V9R 5S5 Canada

Version 0.2

1 Introduction

cew: is a C++ component exerciser workbench. Testing C++ components with *cew* involves a scripting language and a driver generator. A *cew* script consists of a C++ program embedded with test cases. *cew* is oriented towards highly automated testing, where the driver invokes C++ component operations and checks returned values and signaled exceptions. The component-under-test fails a test case if actual behaviour deviates from expected behaviour.

Section 2 of this paper introduces an integer-set abstract data type (ADT) called `IntSet` and its testing infrastructure based on *cew*. Section 3 deals with normal-behaviour testing and Section 4 deals with exception testing. *cew*'s modes of operation are covered in Section 5. Section 6 overviews how to get *cew* and how to invoke it within the context of `IntSet`. Section 7 discusses *cew* limitations and known bugs.

2 Systematic Testing

A systematic approach to testing requires that testing be planned, documented and maintained. For testing to be effectively maintained, it must be based on a maintainable testing infrastructure. *cew* provides a maintainable testing infrastructure for C++ components.

To illustrate the overall structure of *cew*, we introduce `IntSet`, an integer-set ADT written in C++. Figure 1 shows `IntSet`'s interface specification. It is contained in the file `IntSet.h` and includes a class definition, and documentation for all the functions associated with the ADT. Figure 2 contains a *cew* script for `IntSet`. It is contained in the file `bats.script`. The script contains five test cases. The following overviews *cew* constructs contained in `bats.script`:

- `include(CewDir/bin/cew.c++)` includes *cew* macro definitions for use within the script.
- `cew_Start_Menu ..` menu specification is concerned with interactive testing and is discussed in Section 5.
- `cew_Start_Exception_Handler_Builder ..` exception handler specification is concerned with exception testing and is discussed in Section 4.
- `cew_Set_Mode(..)` sets the *cew* mode of operation. `cew` modes are discussed in Section 5.
- `cew_Ncase(..)` is concerned with normal behaviour testing and is discussed in Section 3.
- `cew_Ecase(..)` is concerned with exception testing and is discussed in Section 4.
- `cew_Summary` generates summary statistics after the script is executed. For example, the `bats.script` script yields the following summary:

```
**********Summary**********
Total number of test cases = 4
Total number of test cases in error = 3
```

3 Normal Behaviour Test Cases

Normal-behaviour testing is achieved with *cew* using a `cew_Ncase` test case. The syntax is *cew_Ncase(trace, actval, expval)*. A *trace* is any manipulation of the ADT through its public

```
// The IntSet (integer set) ADT provides access to a set of at most MAXSIZE integer elements.
//      State:
//              s: set of integers
//      Assumptions:
//              none

// Exception Classes
class DuplicateExc {};
class FullExc {};
class NotFoundExc {};

// IntSet Class
class IntSet {
public:
    // Max size of set
    const int MAXSIZE = 3;

    // Assumptions:
    //    none
    // Behaviour:
    //    instance an empty set
    IntSet();

    // Assumptions:
    //    none
    // Behaviour:
    //    if x is an element of the set then
    //       throw DuplicateExc
    //    else if the set is full then
    //       throw FullExc
    //    else
    //       add x to the set
    void add(int);

    // Assumptions:
    //    none
    // Behaviour:
    //    if x is not an element of the set then
    //       throw NotFoundExc
    //    else
    //       delete x from the set
    void delete(int);

    // Assumptions:
    //    none
    // Behaviour:
    //   if x is an element of the set then
    //       return true
    //   else
    //       return false
    bool isMember(int);

protected:
    // ...
};
```

Figure 1: `IntSet` Interface

```
#include <iostream.h>
#include <string.h>
#include "IntSet.h"

include(SpeDir/bin/cew.c++)

cew_Start_Menu
   cew_Menu_Item(a,Add to s, int x;cout << "Enter Integer: ";cin >> x;s.add(x))
   cew_Menu_Item(d, Delete from s, cout << "Not available yet" << endl)
   cew_Menu_Item(m, Check Membership,
   int x;cout << "Enter Integer: ";cin >> x;cout << "Value returned: " << s.isMember(x) << endl)
cew_Stop_Menu

cew_Start_Exception_Handler_Builder
   cew_Build_Handler(DuplicateExc)
   cew_Build_Handler(FullExc)
   cew_Build_Handler(NotFoundExc)
cew_Stop_Exception_Handler_Builder

int main()
{
   cew_Set_Mode(cew_Interactive_On_Failure)

   {IntSet s; cew_Ncase(s.add(1); s.add(2), s.isMember(1), false)}
   {IntSet s; cew_Ncase(s.add(1); s.add(2), s.isMember(1), true)}
   {IntSet s; cew_Ncase(s.add(1); s.add(2); s.add(2), s.isMember(1), true)}
   {IntSet s; cew_Ecase(s.add(1); s.add(2); s.add(2), FullExc)}

   cew_Summary
}
```

Figure 2: *cew* Script (`bats.script`)

interface. The *actval* is an expression that is evaluated after the trace. Its value is taken as the "actual value" of the trace. *expval* is the value that *actval* is expected to have. *cew* reports an error if *actval* is not equal to *expval*.

4 Exceptional Behaviour Test Cases

Exception testing is achieved with *cew* using a `cew_Ecase` test case. The syntax is *cew_Ecase(trace, expexc)*. *expexc* is the exception the *trace* is expected to throw. *cew* reports an error if no exception is thrown or if an exception other than *expexc* is thrown.

cew must be made aware of ADT exception names such that exception handlers may be constructed. To this end, exceptions must be specified using `cew_Build_Handler`. The syntax is *cew_Build_Handler(exc)*. *exc* is the name of the exception that may be thrown by the ADT. Typically, there is one instance of `cew_Build_Handler` for each exception that the ADT may throw. A block of `cew_Build_Handlers` is delimited using `cew_Start_Exception_Handler_Builder` and `cew_Stop_Exception_Handler_Builder` (see example in Figure 2).

5 Operation Modes

cew operates in one of three modes *viz*, `cew_Interactive`, `cew_Interactive_On_Failure` and `cew_Batch`. `cew_Batch` mode is used for regression testing. In `cew_Batch` mode, *cew* produces a listing detailing failed test cases and summary statistics. The other two modes allow for interactive testing. In interactive testing, a menu-driven interface to the ADT is provided. A menu is composed of menu items. *cew* must be made aware of menu items and their associated actions. To this end, menu items must be specified using `cew_Menu_Item`. The syntax of a

menu item is *cew_Menu_Item(sel_char, prompt, action)*. *sel_char* is used to select a menu item and *prompt* describes the action associated with a menu item. On selection, a menu item's *action* is executed. This results in manipulation of the ADT through its public interface.

Typically, there is one instance of `cew_Menu_Item` for each access routine or method supported by the ADT. A block of `cew_Menu_Items` is delimited using `cew_Start_Menu` and `cew_Stop_Menu` (see example in Figure 2).

The interactive tester is executed after the execution of every test case in `cew_Interactive` mode and after the execution of a failed test case in `cew_Interactive_On_Failure` mode. For example, on executing the script in `bats.script` the interactive tester is invoked after executing the first test case. The menu displayed is:

```
FAILURE (Ncase) in test number 1
 Initial test trace = s.add(1); s.add(2)
 Actual value = 1
 Expected value = 0
 Actual expression = s.isMember(1)
 Expected expression = false
 Source script line number = 25


 a: Add to s
 d: Delete from s
 m: Check Membership
 q: Quit

Enter menu selection:
```

6 *cew* Usage

cew and `IntSet` have been bundled together. A tar file can be found at:

`http://www.engr.uvic.ca/~seng422/assignments.html`.

After downloading `IntSet.tar`, un-tar it by executing `tar xvf IntSet.tar`. This should

produce the directory `IntSet.cew2` in which you will find to sub-directories `eg1` and `eg2`. The subdirectories each contain ian example of a *cew* scripts. The example shown in this paper can be found in `eg2`. Locate to either directory. To produce the executable driver enter the command `make bats`. To execute the driver enter `bats`. Other makes-targets are detailed in the `Makefile`.

7 Limitations and Bugs

1: Only one test case is allowed per line in a *cew* script.

2: If the ADT does not define any exceptions, *cew* still requires the following empty exception handler builder block:

```
cew_Start_Exception_Handler_Builder
cew_Stop_Exception_Handler_Builder
```

3: If no interactive testing is to be performed, *cew* still requires the following empty menu builder block:

```
cew_Start_Menu
cew_Stop_Menu
```

4: *cew* suffers from code-bloat.

5: *cew* requires a `m4`/Unix environment.

6: *cew* constructs (such as `cew_Ncase`) inside a C++ comment are expanded. This can cause problems if the C++ comment `//` is used as the construct may expand to more than one line and hence outside the scope of the comment.

7: There are no known bugs.