Subject: unit testing for IDL?
Posted by Thomas Pfaff on Wed, 13 Apr 2005 13:14:24 GMT
View Forum Message <> Reply to Message

Hello,

just curious if there is anything like a unit testing framework around
for IDL programs (like junit, or python's unittest module), or is
everybody testing his/her object oriented programs with his/her own methods?

Thanks for any information.

Thomas Pfaff

Subject: Re: Unit Testing
Posted by Michael Galloy on Wed, 03 Jan 2007 20:28:11 GMT
View Forum Message <> Reply to Message

I have done this a few times. I should probably spruce up the last
version and make some documentation. I have also wanted to build a GUI
test runner (with hopes of someday integrating it into the new
Eclipse-based DE).

What you have looks reasonable, my questions would be:

1. Why isn't it object-oriented?
2. You say "The final aim of this project is to fully support xUnit
testing automation, including support for fixtures." How will you
support fixtures?
3. How does "unitException" have access to local variables?
4. Why do the test names end in an ordinal? Why not
hashtable__testAdding, etc?

Mike
--
www.michaelgalloy.com

Subject: Re: Unit Testing
Posted by Robbie on Wed, 03 Jan 2007 22:51:52 GMT
View Forum Message <> Reply to Message

Dear Mike,

Thanks for the feedback. I think my key problem is that I haven't used

unit testing in other languages before. I would like to make something which compliments the incremental compiling nature of IDL and I want to avoid writing a .pro file parser.

> 1. Why isn't it object-oriented?
I think that writing objects in IDL is quite clumsy. I guess I haven't had the need for unit tests to be based inside objects yet. I'm concerned that using OO would deviate from unit tests being short and sweet.

> 2. You say "The final aim of this project is to fully support xUnit
> testing automation, including support for fixtures." How will you
> support fixtures?
I was thinking of getting fixtures to SetUp() and TearDown() a common block. I could also use keywords to do the same sort of thing. This is where I should probably be using OO.

> 3. How does "unitException" have access to local variables?
I can't believe I missed that one! I should probably stick to using wrappers of CALL_PROCEDURE, CALL_FUNCTION and CALL_METHOD. unitExceptionFunction just doesn't roll off the tongue too well :)

> 4. Why do the test names end in an ordinal? Why not
> hashtable__testAdding, etc?
I've developed a nasty habit of using ordinals in the suffix. I guess I shouldn't tempt anyone else to do the same thing. Any procedure, function or method with __test in it would become a unit test. Perhaps I should allow unitSearch specify exclusions.

Thanks

Robbie

Subject: Re: Unit Testing
Posted by Michael Galloy on Wed, 03 Jan 2007 23:16:46 GMT
View Forum Message <> Reply to Message

Robbie wrote:
> Thanks for the feedback. I think my key problem is that I haven't used
> unit testing in other languages before. I would like to make something
> which compliments the incremental compiling nature of IDL and I want to
> avoid writing a .pro file parser.
>
>> 1. Why isn't it object-oriented?
> I think that writing objects in IDL is quite clumsy. I guess I haven't
> had the need for unit tests to be based inside objects yet. I'm
> concerned that using OO would deviate from unit tests being short and

> sweet.

I'm including an example test (a rather silly one, for instructional purposes only) at the bottom of this post. I don't think it has much "extra fat."

>> 2. You say "The final aim of this project is to fully support xUnit
>> testing automation, including support for fixtures." How will you
>> support fixtures?
> I was thinking of getting fixtures to SetUp() and TearDown() a common
> block. I could also use keywords to do the same sort of thing. This is
> where I should probably be using OO.

I definitely like objects for this.

>> 3. How does "unitException" have access to local variables?
> I can't believe I missed that one! I should probably stick to using
> wrappers of CALL_PROCEDURE, CALL_FUNCTION and CALL_METHOD.
> unitExceptionFunction just doesn't roll off the tongue too well :)

I have a couple batch files that have error handling in them. Put "@error_is_pass" in your test if the test is supposed to cause an error.

>> 4. Why do the test names end in an ordinal? Why not
>> hashtable__testAdding, etc?
> I've developed a nasty habit of using ordinals in the suffix. I guess I
> shouldn't tempt anyone else to do the same thing. Any procedure,
> function or method with __test in it would become a unit test. Perhaps
> I should allow unitSearch specify exclusions.

No big deal, I'm going to try to polish what I have up a bit (and add some documentation) and get it posted on my website soon. I'll let you know when it's up.

Mike
--
www.michaelgalloy.com

Here are the tests:

```
;+
; This test fails because the assertion is wrong.
;-
function findgentest::test1
  a = findgen(5)
  assert, n_elements(a) eq 6, 'Wrong number of elements'
```

```
  return, 1
end


;+
; This test should pass the assertion and return 1 (i.e. success).
Tests can
; also return 0 or generate an error to indicate failure.
;-
function findgentest::test2
  a = findgen(5)
  assert, array_equal(a, [0.0, 1.0, 2.0, 3.0, 4.0]), 'Correct elements'

  return, 1
end


;+
; This is a test that will pass because the code of the test is
supposed to
; cause an error. To do this kind of test, use the "error_is_pass"
batch file.
;-
function findgentest::test3
  @error_is_pass

  a = findgen('string')

  return, 1
end


;+
; This is a test that will fail on an io error because of the use of
the
; "error_is_fail" batch file. IO errors don't normally cause a test to
fail.
;-
function findgentest::test4
  @error_is_fail

  a = findgen('another_string')

  return, 1
end
```

```
;+
; Inherit from MGtestCase.
;
; @file_comments To create a test case just inherit from MGtestCase and
create
;               method with names that start with "test". This test
can be run
;               with the command: mgunit, cases='findgentest'
;-
pro findgentest__define
  define = { findgentest, inherits MGtestCase }
end
```

Subject: Re: Unit Testing
Posted by Richard French on Thu, 04 Jan 2007 02:06:30 GMT
View Forum Message <> Reply to Message

Could you supply a reference for Unit Testing for those of us who are not
familiar with it?
Thanks,
Dick French

On 1/2/07 9:14 PM, in article
1167790448.349519.240060@h40g2000cwb.googlegroups.com, "Robbie"
<retsil@iinet.net.au> wrote:

> Hi,
>
> Are there any implementations of Unit Testing in IDL yet? I've put
> together a white paper for unit testing in IDL. It is fairly straight
> forward, and I've tried to make it easy to run tests of currently
> compiled code. However, there is no point implementing it if someone
> has a better idea of how to go about unit testing.
>
> http://www.barnett.id.au/idl/UnitRun.html
>
>
> Robbie
>
> ---------- ----------
> ----------
>
> UnitRun is an adaptation of testing frameworks such as NUnit, JUnit and
> PyUnit. The final aim of this project is to fully support xUnit testing
> automation, including support for fixtures.

> Unit test procedures
> Unit tests are called using specially named test procedures.
> Expected procedure names:
>
>   * hashtable__test
>   * hashtable__test0
>   * hashtable__test1
>   * hashtable__test0000445
>
> A simple test case
>
> unitSearch
>
> pro hashtable__test1
> obj = obj_new('hashtable')
> obj -> Add, 'one', 1
> unitAssert, obj -> isContained('one')
> obj_destroy, obj
> end
>
> The unitSearch directive indicates that all subsequent test procedures
> should be included as unit tests. The unitAssert procedure reports the
> result of the test.
>
>   1. The unitAssert procedure reports the name of the test procedure
> (see HELP, CALLS=calls)
>   2. The unitAssert procedure reports <success> if the argument is
> greater than one
>   3. The unitAssert procedure reports <fail> in any other circumstance
> including unhandled execptions
>
> Running tests
>
> reslove_routine, 'hashtable__test1'
> unitRun
>
> The unitRun procedure looks for all specially named test procedures.
> All test procedures are re-resolved. Only procedures with the
> unitSearch directive will be included in unit test.
>
> unitRun, ['hashtable__test1','hashtable__test2']
>
> The unitRun procedure can be used to manually call a sequence of test
> procedures. In this case procedures are called directly without paying
> attention to unitSearch.
>
> unitRun, LOG_FILE='unitRun.log'
>

> The unitRun procedure can dump the test results to a log file instead
> of dumping the results to the IDL command line.
> Expected exceptions
>
> pro hashtable__test2
> obj = obj_new('hashtable')
> obj -> Add, 'one', 1
> unitException, 'obj -> Add, 5, 6'
> obj_destroy, obj
> end
>
> The unitException procedure executes a single IDL statement which is
> expected to result in an exception. The unitException does not
> currently distinguish between message blocks or names.
>
> 1. The unitException procedure reports <success> if an exeception
> was encountered
> 2. The unitException procedure reports <fail> if no exception was
> encountered
>

Subject: Re: Unit Testing
Posted by Robbie on Thu, 04 Jan 2007 02:38:41 GMT
View Forum Message <> Reply to Message

Historically, unit testing has been around for a long time. It is simply the act of writing a program to test your program.

http://en.wikipedia.org/wiki/Unit_testing refers to one of the original standards.
"IEEE Standard for Software Unit Testing: An American National Standard, ANSI/IEEE Std 1008-1987"

Nowadays programmers expect unit testing to be integrated into the language and/or IDE. JUnit/Eclipse was the first popular package to implement this kind of unit testing. Just have a look at http://junit.sourceforge.net/doc/cookbook/cookbook.htm to get a synopsis of how the syntax is in Java.

Another synopsis is at http://www.xprogramming.com/testfram.htm

Wikipedia says that "The overall design of xUnit frameworks depends on several components.". I suspect that there are many possible designs for unit testing in IDL, hence the purpose of this thread.

Subject: Re: Unit Testing
Posted by Robbie on Thu, 04 Jan 2007 02:51:33 GMT
View Forum Message <> Reply to Message

Matlab has a few unit testing packages as well

http://mlunit.sourceforge.net/index.php/Unit_Testing_With_Ma tlab

Robbie

Subject: Re: Unit Testing
Posted by Michael Galloy on Fri, 05 Jan 2007 04:51:46 GMT
View Forum Message <> Reply to Message

OK, I documented the code, made a few (admittedly silly) examples, and
wrote a bit of explanation up. If you're interested it's at:

http://michaelgalloy.com/2007/01/04/unit-testing-framework.h tml

Mike
--
www.michaelgalloy.com

Subject: Re: Unit Testing
Posted by Qing on Wed, 07 Feb 2007 02:10:44 GMT
View Forum Message <> Reply to Message

On Jan 5, 3:51 pm, Michael Galloy <mgal...@gmail.com> wrote:
> OK, I documented the code, made a few (admittedly silly) examples, and
> wrote a bit of explanation up. If you're interested it's at:
>
> http://michaelgalloy.com/2007/01/04/unit-testing-framework.h tml
>
> Mike
> --www.michaelgalloy.com

Hi Mike,

These are very interesting stuff. How practical do you think this can
be applied to a real program?
For example, one writes a routine accepting a list of parameters and
keywords. Inside the routine,
there are many calls to other native IDL libraries and user routines,
and the parameters/keywords
passed can include constarts/arrays/structures/pointers/objects...
Does anyone know if all native IDL libraries have been tested this

way?
Qing