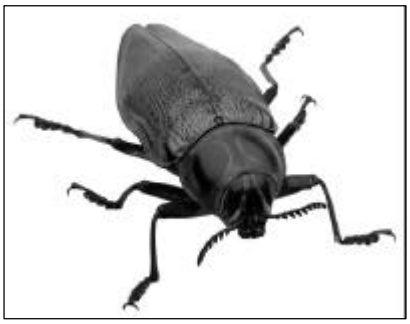
The Ptolemy II Test Bed



Christopher Hylands
<cxh@eecs.berkeley.edu>

The Ptolemy II Test Bed

- Regression testing
- Nightly Builds
- Scripting is good - easy access to the code, fast development of tests
- Jacl or Tcl Blend - Interfaces between Tcl (a scripting language) and Java (a system language)
- Code coverage tools provide verification
- Todo: Performance Measurements
  GUI Testing

Why Test?

- Makes development easier - changes that break the code are quickly detected
- Shipping product is easier - we've been testing all along
- Poorly tested code is usually incorrect code
- Developing code is not just writing code: design, testing and maintenance usually take up more time.

Nightly Builds

- Happens every night, email is sent to the group
- "Don't break the build" - prompts developers to test changes before checking them in
- Developers see problems immediately
- Build a distribution every night - When we ship, much of the work is done

Scripting

- Using Scripting to write tests for Java is quick and easy
- Writing tests is much more of an incremental process than writing system code - a scripted language makes sense
- Being able to easily modify tests, and then run them from an interpreter makes test case development faster

Jacl and Tcl Blend

- Jacl and Tcl Blend provide an interface between Tcl (a scripting language), and Java, (a system language)
- Jacl - An implementation of Tcl written solely in Java.
- Tcl Blend - A platform dependent Tcl extension that gets loaded into Tcl
- So what's the difference?

Jacl

- First Implemented by Ioi Lam while at Cornell
- 100% Java implementation of most of the Tcl 8.x interpreter
- Main Benefit: Platform independent, can be used in applets
- Main Drawback: Can be very slow, especially for recursion

Tcl Blend

- First Implemented by Ken Corey and Scott Stanton while at Sun Microsystems
- Tcl extension that gets loaded into a Tcl Program like tclsh or wish
- Main Benefit: Provides easy access to Java code to preexisting Tcl Programs
- Main Drawback: Platform dependent - Currently runs under 95/98/NT, Solaris, Linux, Digital Unix

Access to Java

- Jacl and Tcl Blend provide the same interface between Tcl and Java
- Tcl command to instantiate a Java object: `set a [java::new` *`classname`*`]`
- Returns a handle, like *`java0x4`*
- We can then call Java methods on the handle:
  `$a toString`

Create a Java String

- Create a Java String, get its value

```
% set a [java::new \
                {String String} \
                "A string"]
java0x4
% $a toString
A string
```

Tcl Testing Framework

- First implemented by Mary Ann May-Pumphrey of Sun Microsystems
- Create a Tcl proc called `test`
- Usage:

```
test testname {comment} {
 # code to run
} {expected results}
```


A Simple Test

```
test SimpleTest-1.1 {Test Foo} {
  set a \
     [java::new {String String} \
                "A string"]
  $a toString
} {A string}
```

An Actual Ptolemy II Test

```
test NamedObj-2.1 {Create a NamedObj, \
                   set the name, change it} {
    set n [java::new \
                  ptolemy.kernel.util.NamedObj]
    set result1 [$n getName]
    $n setName "A Named Obj"
    set result2 [$n getName]
    $n setName "A different Name"
    set result3 [$n getName]
    $n setName {}
    set result4 [$n getName]
    list $result1 $result2 $result3 $result4
} {{} {A Named Obj} {A different Name} {}}
```


Code Coverage

- Run the test suite and use a tool to measure code coverage
- We use JavaScope from Sun - available at no cost to schools, $795/license otherwise
- 100% code coverage does not mean the code is completely tested
- However, a high level of code coverage is a start

What's Missing

- Formalized timing performance measurements
- Testing the GUI
- Better testing of the interaction between components