Notes on ESpec

What is ESpec?

- A collection of classes that help organize the test cases for a program
- A user adds test cases to the appropriate structure
- The **ESpec** sub-system runs the test cases
- A report is created in one html file for each test class and one html file that combines all the reports of all the test classes.
- You read the reports with a browser.

Installing ESpec in Windows

Be sure to install Eiffel 6.6 first

- Go to the following web page
  http://www.cse.yorku.ca/~sel/espec/download.htm
- Click on [**Here**] on the first line
  - » **with label "Espec (Simplified without GUI)"**
- This downloads a zip file that you unzip producing the folder **espec_simple**
- Move the folder to where you have your Eiffel projects.
- Set an environment variable **espec** to point to the folder **espec_simple**
  - » **--> System control panel --> Advanced tab, and select the button Environment Variables**

Example use of ESpec – part 1

- All tests in one class

```
class ROOT_CLASS
inherit ES_TEST
create make
feature
  make
    do
      add_boolean_case (agent bool1)
      add_violation_case (agent viol1)
      add_violation_case_with_tag (“tag”, agent viol1)
      run_espec
    end
…
```

Example use of ESpec – part 2

- Have one **TEST_CLASS** for every class you want to test

```
class TEST_CLASS
   inherit  ES_TEST

create make

feature -- Test class creation
  … See slide  “Feature -- Test class creation”

feature -- Test cases
  … See slide  “Feature -- Test cases”

end
```

Feature -- Test class creation

```
make
  -- Defines the test cases to use
  do
      add_boolean_case (agent test_junk)
      add_violation_case (agent test_precondition)
      add_violation_case_with_tag
            (“valid_bounds” , agent test_precondition)
end
```

Feature -- Test cases creation

```
test_junk : BOOLEAN
        do
            comment("test_junk")
            Result := true        -- Try with false
        end

test_precondition
        local array : ARRAY[STRING]
        do
            comment("test_precondition")
            create array.make(5,1)   -- Try with make(1,5)
                    -- Try after changing, on previous slide,
                    -- “valid_bounds” to “wrong_tag”
        end
```

Putting many test classes together

- Have one **ROOT_TEST** for the system

```
class ROOT_TEST
   inherit  ES_SUITE
create  make
feature make
         do
            add_test(create {TEST_CLASS_1}.make)
            add_test(create {TEST_CLASS_2}.make)
            show_browser
            run_espec
         end
```

Note different inherit class

One for each class to test

Automatically starts the default browser

For later runs need to reload the page

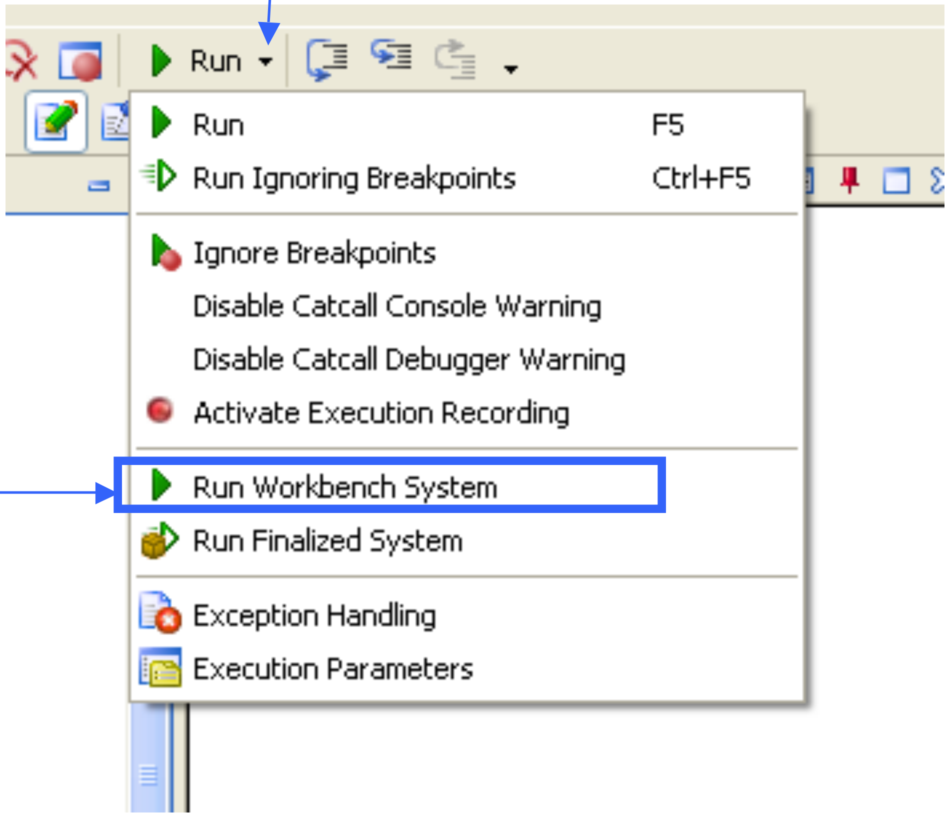
Running the tests – method 1

Press and release – slower than a click – the arrow beside the Run button to get the menu

The system will run when Run is clicked BUT it will stop at every false assertion.

Need to keep clicking Run to continue execution – a PAIN

Select

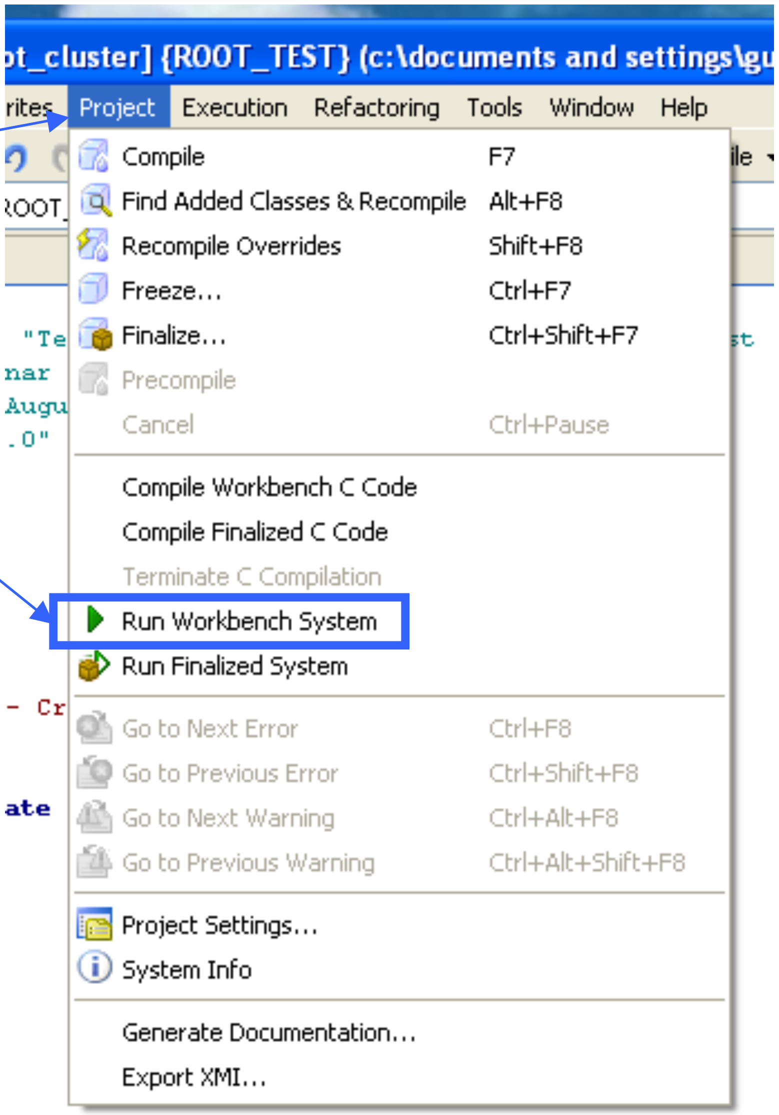


Run
Run F5
Run Ignoring Breakpoints Ctrl+F5
Ignore Breakpoints
Disable Catcall Console Warning
Disable Catcall Debugger Warning
Activate Execution Recording
Run Workbench System
Run Finalized System
Exception Handling
Execution Parameters

Running the tests – method 2

Click the Project button to get the menu

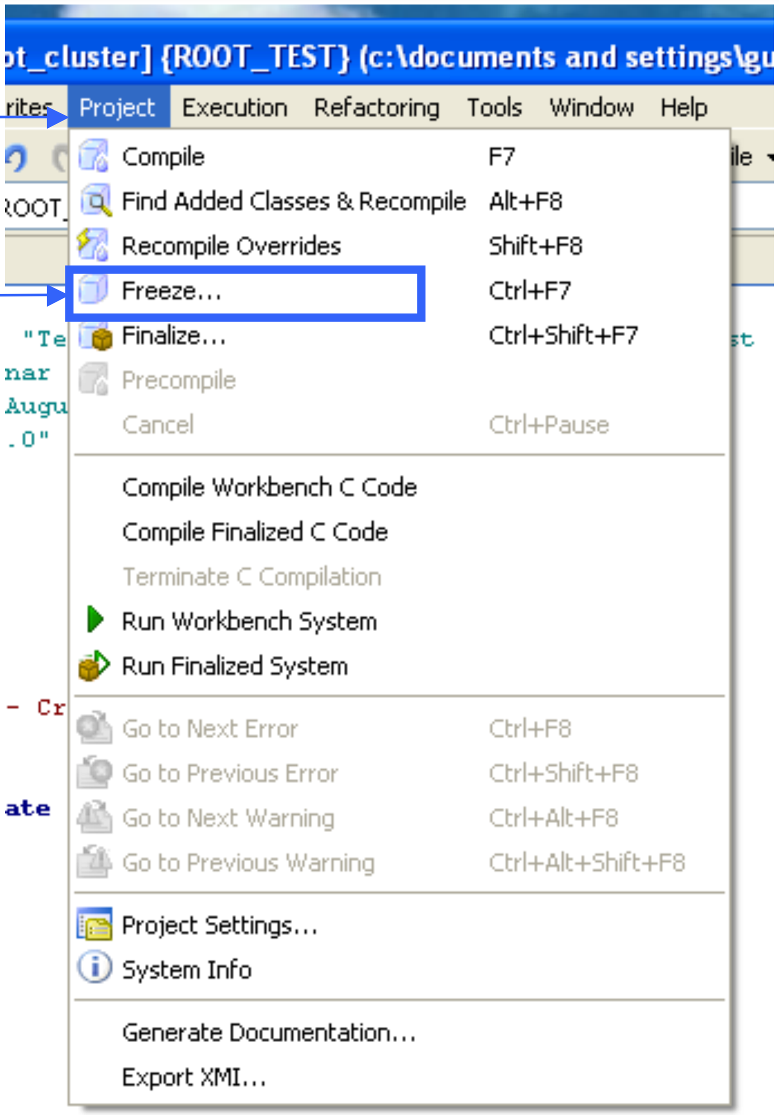
Select



Freeze'ing the system

Click the Project button to get the menu

Select



Sometimes, when new agents are created, the system will terminate with an error message. In that case you freeze the system.