Testing
(in EiffelStudio using CDD)

Andreas Leitner, Arno Fiva, Stefan Mori

ETH Zurich, Switzerland
Overview

- Testing crash course
- Special EiffelStudio version: "Contract Driven Development"
- Study to evaluate use of testing
Kinds Of Tests

- Unit test
- Integration test
- System test
- Acceptance test
- Regression test
Unit Tests

class BANK_ACCOUNT
feature
  balance: INTEGER
  deposit (v: INTEGER)
    do
      balance := balance + v
    end
end
class TEST_BANK_ACCOUNT
inherit TEST_CASE
feature
  test_deposit
    local
      ba: BANK_ACCOUNT
    do
      create ba
      ba.deposit (30)
      assert (ba.balance = 30)
    end
end
Test First Development

1. Add tests
2. Run tests
3. Pass?
   - yes
   - no
4. Add / Change Code
5. Pass?
   - yes
   - no
6. Run tests
Contract Driven Development (CDD)

- CDD is modification of EiffelStudio
- CDD must be used for Project
- Available from http://dev.eiffel.com/CddBranch
- Adds support for testing to IDE
Contract Driven Development
Instalation

- Windows
  - Install Visual C++ or Platform SDK first
  - Then use installer

- Linux
  - Untar
  - Set/update environment variables PATH, ISE_EIFFEL, ISE_PLATFORM
Contract Driven Development (CDD)

Manually Written Test Cases (xUnit style)

Extracted Test Cases (from program runs)
class BANK_ACCOUNT
feature
  balance: INTEGER
  deposit (v: INTEGER)
    do
      balance := balance + v
    end
end

class TEST_BANK_ACCOUNT inherit TEST_CASE
feature
  test_deposit
    local
      ba: BANK_ACCOUNT
    do
      create ba
      ba.deposit (30)
      assert (ba.balance = 30)
    end
end
Contracts

```plaintext
class BANK_ACCOUNT
feature
    balance: INTEGER
    deposit (v: INTEGER)
        require
            v > 0
        do
            balance := balance + v
        ensure
            balance < old balance
            balance = old balance + v
        end
    invariant
        balance >= 0
end
```

- Monitored at runtime
- Violations stop debugger
class BANK_ACCOUNT
feature
  balance: INTEGER

  withdraw (v: INTEGER)
    require
      v > 0
    do
      balance := balance - v
    ensure
      ...
    end
  ...

invariant
  balance >= 0
end

class MAIN_WINDOW
feature
  ba: BANK_ACCOUNT
  field: TEXT_FIELD

  on_withdraw
    local
      v: INTEGER
    do
      v := field.as_integer
      ba.withdraw (v)
    end
end
Extracted Test Case

Snapshot of relevant state taken
Works with primitive and composite objects

withdraw (v: INTEGER)

require
v > 0

do

ensure
balance < old balance
balance = old balance + v

end

invariant
balance > 0

extracted_test_case
do
ba := new_object ("BANK_ACCOUNT")
set_field (ba, "balance", 300)
check_invariant (ba)
ba.withdraw
end

v 30
balance 300
Whose fault is it?

```plaintext
extracted_test_case
do
    ba := new_object ("BANK_ACCOUNT")
    set_field (ba, "balance", 300)
    check_invariant (ba)
    ba.withdraw (20)
end
```

**Postcondition violation**

**Precondition violation**

- `event_loop
  GUI
c_0`
- `TEXT_FIELD
c_1`
- `set
  TEXT_FIELD
c_2`
- `as_integer
  TEXT_FIELD
c_4`
- `on_withdraw
  MAIN_WINDOW
c_3`
- `withdraw
  BANK_ACCOUNT
c_5`
Demo
When things don't work

- Unresolved test cases
  - Changes in system under test
  - When extraction fails
- Re-Extraction
Evaluation

- Conducting study to evaluate use of testing
- Does not affect course, data treated anonymously
- CDD-Eiffelstudio produces testing logs
- Data stored in directory `cdd_tests`
- Commit directory to svn (once a week)
- If logs (*.log) become too large, zip old ones
- Questions? Talk to me or assistant right now
Contract Driven Development (CDD)

- [link] http://dev.eiffel.com/CddBranch
- Use this version for project
- New release almost every week
- Check frequently and install updates
- Commit logs once a week