

Introduction To MockObjects

Tim Bacon, Duncan Pierce
Thames Valley Agile SIG

What is a MockObject?

- Representation of the behaviour / state of the “rest of the system”
 - Bounds the class / component under test
- Observer of method calls / results
 - Tests expectations using “expect - do - verify” pattern
- An instance of a required interface
 - domain and infrastructure level mocks
 - e.g. MockConnection, MockCreditApprovalStrategy

Why do we need MockObjects?

- Isolates the tested class / component
 - for fine-grained (accurate!) error detection
- Enables tests for special behaviour
 - e.g. file / network / database not present
 - Exception handling is usually untested!
- Promotes good (test-driven) design
 - fits the "system composition" / "methods as messages" / "tell don't ask" paradigms

MockObjects in practice...

- Design tends towards small classes with specific responsibilities
 - loosely coupled components and systems
 - mocks fit well with CRC techniques
- Design tends to favour composition at creation
 - “dependency injection” constructional pattern
- But: means too many classes? promotes over-confidence? how to handle code generation?

Where can I find out more about MockObjects?

- “Endo-Testing: Unit testing with mock objects”
 - <http://mockobjects.com/MocksObjectsPaper.html>
- Java libraries
 - Sourceforge project “mockobjects”(historical interest)
 - EasyMock (dynamic, strongly-typed library)
 - Jmock (dynamic, "Stringly-typed" library)
- Libraries for many other languages too
 - e.g. .Net, Ruby, Python, C++

Library example

- Current design caters for simple loans
 - Book: loanTo(), title, author, isbn, loanedTo[], copies
 - Person: return(), name, address, booksOnLoan[]
 - DataStore: save()
- Want to add more functionality:
 - e.g. variable-length loans, sending overdue notices, making reservations, adding CDs
 - Need to restructure / extend the design. How? What mocks might we need? For which sort of tests?