Test Driven Development for Embedded Software

James W. Grenning
Embedded Systems Conference
San Jose, April 2007
ESC Class# 241

Copyright © James W. Grenning
All Rights Reserved

Schedule Killers

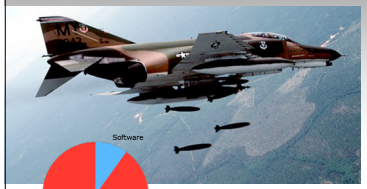
"The only reasonable way to build an embedded system is to start integrating today… The biggest schedule killers are unknowns; only testing and running code and hardware will reveal the existence of these unknowns." [GANSSLE]

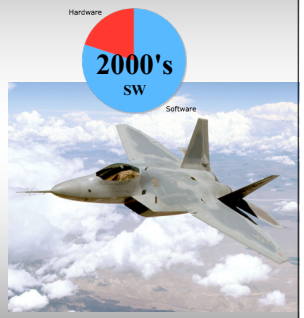
[GANSSLE] Ganssle, Jack, The Art of Designing Embedded Systems, Butterworth-Heinemann, Woburn MA, p.48

Jack goes on to say…

"Test and integration are no longer individual milestones; they are the very fabric of development."

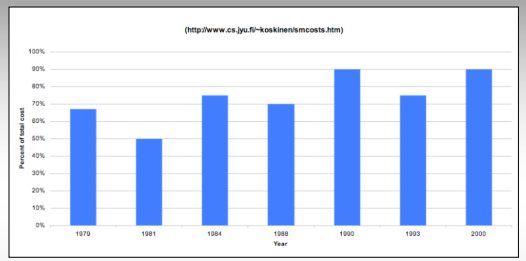
More Functionality in Software





6 years

Software Maintenance Cost



National Institute of Standards and Technologies (NIST)(1) estimates the annual cost of software defects in the United States as $59.5 billion

DoD spent $12B on rework in 2006

http://www.gcn.com/print/25_18/41177-1.html

Copyright © James W. Grenning
All Rights Reserved

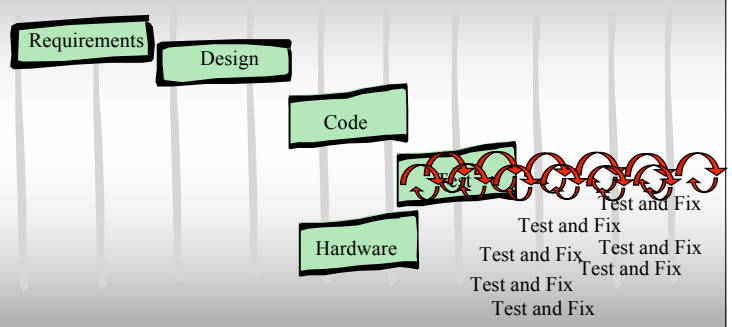
Where is the Time Going?

- 50% to Debug is commonly claimed
- 25% of all defects are introduced while changing and fixing code [R.B Grady, Software Process Improvement]

- One messed up project…
  - 5 months in requirements
  - 3 months development
  - 6 one month test and fix cycles

Copyright © James W. Grenning
All Rights Reserved

Typical Development Cycle



Copyright © James W. Grenning
All Rights Reserved

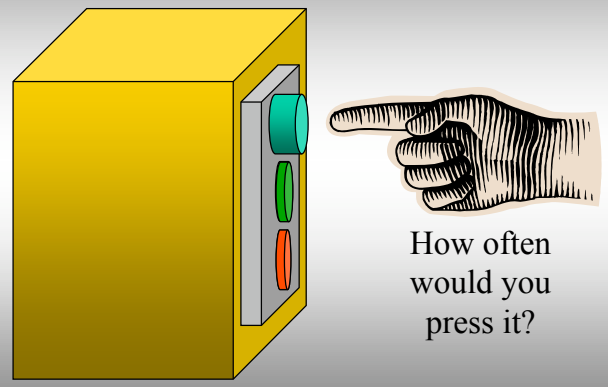
Systems Must be Designed to be Maintained

- Automated test is key
- Systems evolve, the design is NEVER done

- Key technical practices for evolving design
  - Test Driven development
  - Refactoring
  - Modularity, Loose Coupling, High Cohesion (OO Design)

Copyright © James W. Grenning
All Rights Reserved

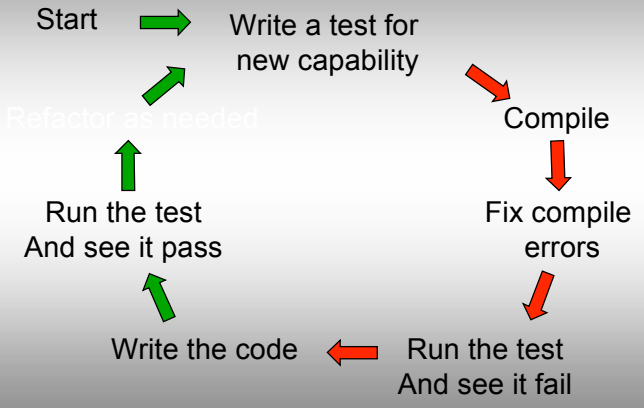
Imagine You Have a Button to Press That Would Tell You If Your Code Worked



How often would you press it?

Copyright © James W. Grenning
All Rights Reserved
www.objectmentor.com
grenning@objectmentor.com

The Test Driven Development Cycle



Copyright © James W. Grenning
All Rights Reserved
www.objectmentor.com
grenning@objectmentor.com

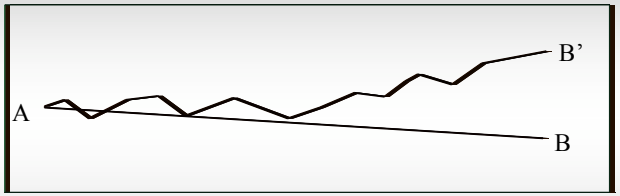
Outer Process

- From the overall architecture, choose the module to work on
- Make a list of tests this module must pass
- Choose a subset of the tests to explore the interface of the module
- Incrementally apply the core TDD cycle to define the interface, the tests are the first client
  - Add more interface tests as needed
- Order the remaining tests such that more complex tests build on the successes of the earlier tests
- Incrementally apply the core TDD cycle to those tests
  - Add more tests as needed
- Check for completeness

Copyright © James W. Grenning
All Rights Reserved
www.objectmentor.com
grenning@objectmentor.com

Lots of Small Steps

- Shortest distance between two points



- Use test-driven to get from A to B in very small, verifiable steps
- You often end up in a better place

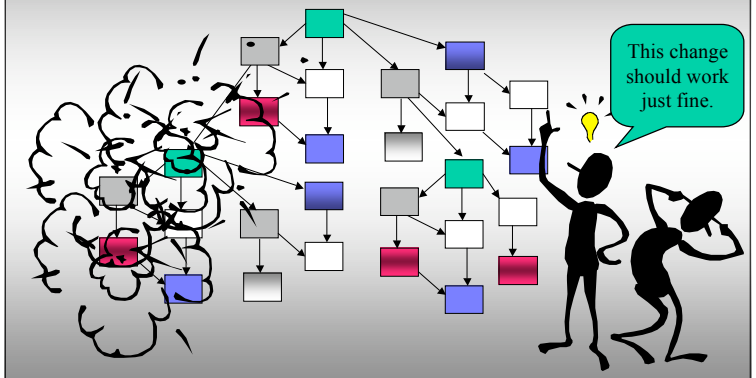
Copyright © James W. Grenning
All Rights Reserved
www.objectmentor.com
grenning@objectmentor.com

Do the Simplest Thing

- Assume simplicity
  - Consider the simplest thing that could possibly work
  - Iterate to the needed solution
- When coding:
  - Take small steps with test providing feedback
  - Build the simplest possible code that will pass the current tests
  - Refactor the code to keep the design simple
  - Eliminate duplication

Copyright © James W. Grenning
All Rights Reserved

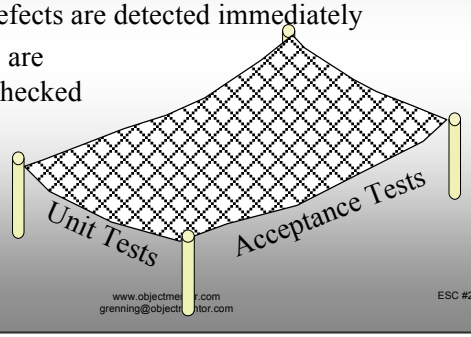
Does this happen to you?



Copyright © James W. Grenning
All Rights Reserved

Automated Tests Support Evolution and maintenance

- Once a test passes, it is re-run with every change
- Broken tests are not tolerated
- Side affect defects are detected immediately
- Assumptions are continually checked



Copyright © James W. Grenning
All Rights Reserved

Automated Tests

- Unit Test
  - Feedback to the developer that the code does what is expected
  - Written using a Unit Test Harness (e.g. CppTestTools, CppUnit, …) [CPPTEST]
- Acceptance Tests
  - Feedback to the *Customer* that the code meets the requirements
  - Written in a domain specific language (e.g. FitNesse) [FITNESSE]
- Tests
  - Provide examples of how to use the code (documentation)
  - Are Automated
  - Run every few minutes, with every change

Copyright © James W. Grenning
All Rights Reserved

Tests give Unambiguous Feedback

All test pass

```
OK (22 tests, 22 ran, 109 checks, 0 ignored, 0 filtered out)
No leaks detected
```

Test failures

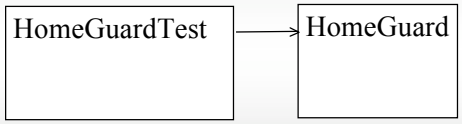
```
Failure in TEST(HomeGuard, CXonfigurationDownload)
    HomeGuardTest.cpp(85)
        expected <COMMUNICATION PROBLEM EXISTS>
        but was  <COMMUNICATION PROBLEM EXITS>

Errors (1 failures, 22 tests, 22 ran, 111 checks, 0 ignored, 0
filtered out)
```


Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

What is Tested?

- Every class (module) has one or more unit tests
- Test everything that can possibly break



Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

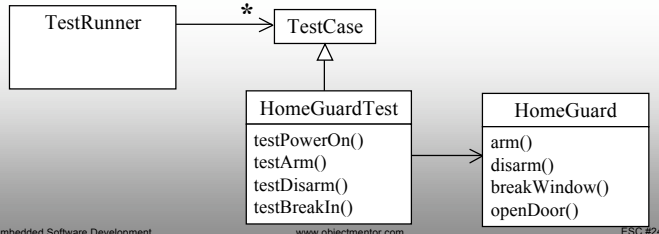
Testing Frameworks

- Tests must be automated
  - Otherwise they won't be run
- Most OO languages have a testing framework, xUnit
  - JUnit, CppUnit(Lite), PyUnit, NUnit, VBUnit
  - A simple tool
  - Collects, organizes and automatically calls your test code
- C can be tested too!

Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

Building Test Classes

- Testing frameworks work similarly
- Your class inherits from a test framework class, allowing your test to be plugged into the Test Runner



Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

CppTestTools

- Free C++ unit test harness
- Uses macros to make test definition easy
- Can be used to test C code
- Tests are written that check binary conditions
- Tests are repeatable

- Download it from
  http://www.objectmentor.com/resources/downloads/index, look for CppTestTools.zip

Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

Example Power-on Test

- This test confirms that the system is initialized into the correct state
  - The system is not armed, it is not making any noise or flashing any lights. The display says the system is READY, and no phone calls have been made

```
TEST(HomeGuard, PowerUp)
{
  CHECK(false == panel->isArmed());
  CHECK(false == panel->isAudibleAlarmOn());
  CHECK(false == panel->isVisualAlarmOn());
  CHECK(panel->getDisplayString() == "READY");
  CHECK(true == phone->isPhoneIdle());
}
```


Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

Example Break-in Test

- This tests confirms that an armed system responds properly to the back window being opened
  - The system is making noise, flashing lights, [pinpointing the intrusion and calling the police

```
TEST(HomeGuard, WindowIntrusion)
{
  homeGuard->arm();
  homeGuard->windowSensorTripped("Back");
  CHECK(true == panel->isArmed());
  CHECK(true == panel->isAudibleAlarmOn());
  CHECK(true == panel->isVisualAlarmOn());
  CHECK_EQUAL("Back Window", panel->getDisplayString());
  CHECK_EQUAL(policePhone, phone->getLastNumberDialed());
}
```


Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

What Else Does the Test Harness Do?

- Prior to test execution, the test harness calls a SetUp function
- The body of the test executes

```
TEST(TestHarness, ExamplePassingChecks)
{
  CHECK(0 == 0);
  LONGS_EQUAL(1,1);
  STRCMP_EQUAL("THIS", "THIS");
  DOUBLES_EQUAL(1.0, 1.0, .01);
}
```

- After the test body the TearDown function is called
- Memory leaks are reported

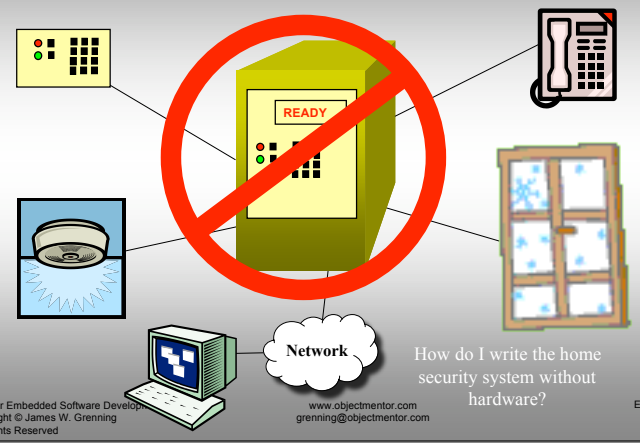
Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

Wait a Minute, You're Talking to Embedded Software Engineers

- We need hardware to test
- We have real time constraints
- Our devices have tiny memories
- It takes 5 minutes to download

Copyright © James W. Grenning
All Rights Reserved
www.objectmentor.com
grenning@objectmentor.com

There's No Hardware

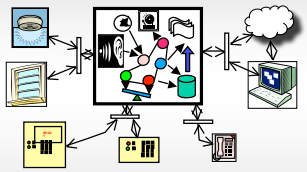


How do I write the home security system without hardware?

Copyright © James W. Grenning
All Rights Reserved
www.objectmentor.com
grenning@objectmentor.com

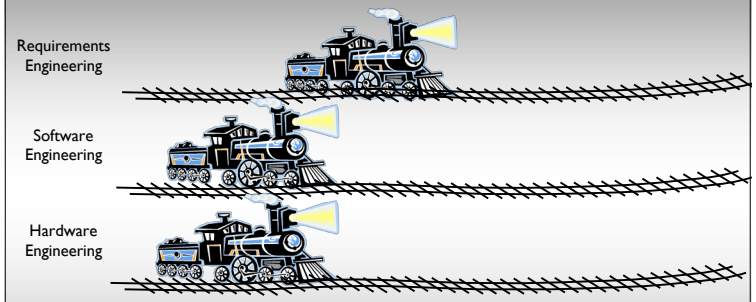
Home alarm system example

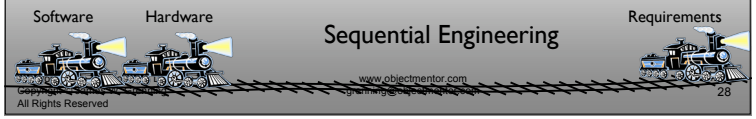
- Front panel with LEDs, push buttons, times square display
- Phone line
- Sensors
- Configuration PC
- The hardware won't be ready for 3 months (two weeks before promised delivery)
- Eval board does not have custom IO



Copyright © James W. Grenning
All Rights Reserved
www.objectmentor.com
grenning@objectmentor.com

Concurrent Engineering





Sequential Engineering

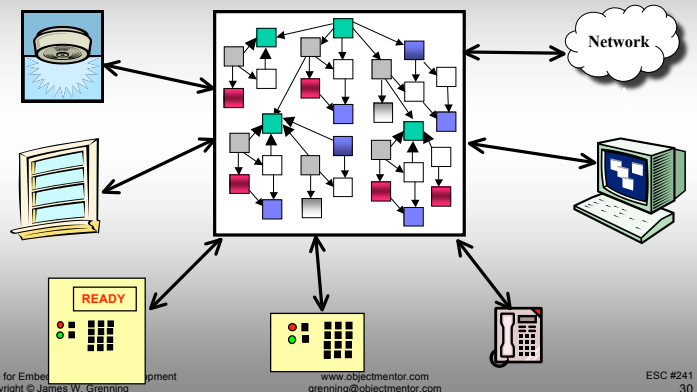
All Rights Reserved
www.objectmentor.com

Code and Tests are Hardware Independent (A lot of it anyway)

- Code is written to be platform independent from day one
- Run tests on the development machine and the target machines
- Hardware API can be "Mocked-out" or stubbed out for test purposes
- Hardware dependencies must be managed (or they will manage you)

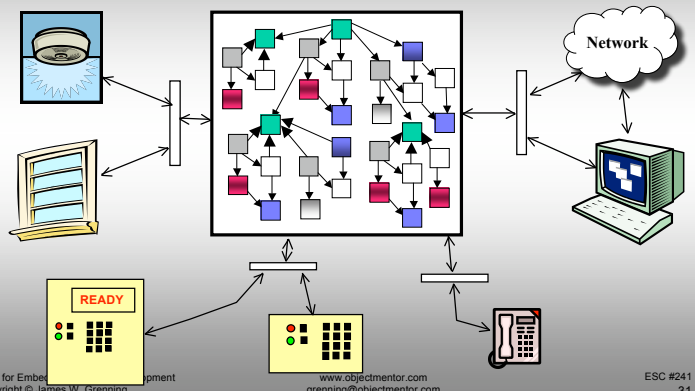
Copyright © James W. Grenning
All Rights Reserved
www.objectmentor.com
grenning@objectmentor.com

Too Often, the Software Depends on Hardware Specifics



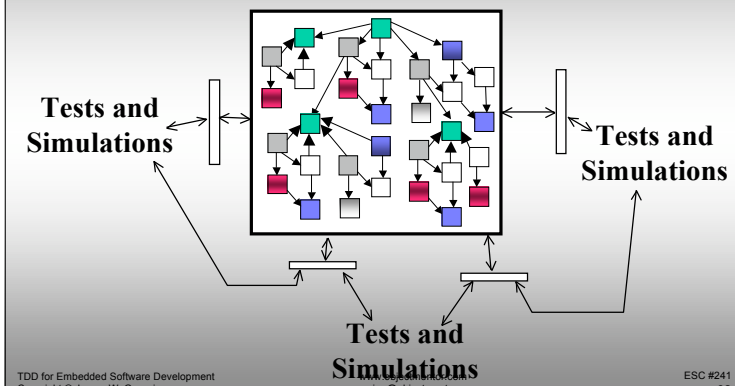
Copyright © James W. Grenning
All Rights Reserved
www.objectmentor.com
grenning@objectmentor.com

Separate Core System Logic from Hardware Specifics using interfaces



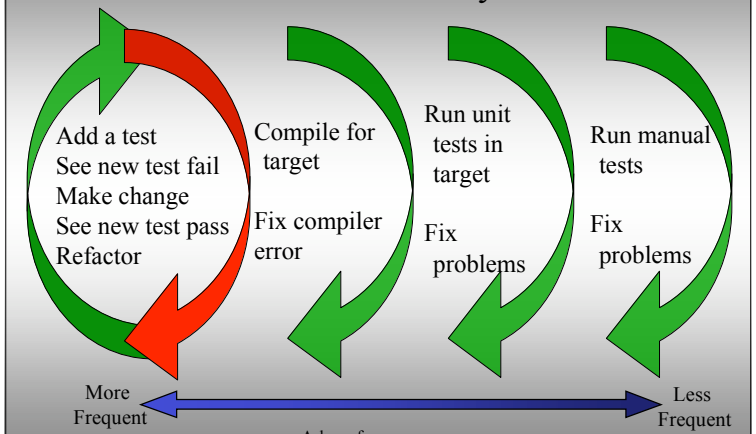
Copyright © James W. Grenning
All Rights Reserved
www.objectmentor.com
grenning@objectmentor.com

Test the Application independently from the Hardware Execution Environment



Copyright © James W. Grenning
All Rights Reserved
www.objectmentor.com
grenning@objectmentor.com

Embedded TDD Cycle



Ask me for my paper

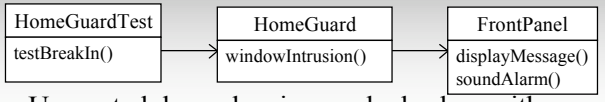
Embedded Test-Driven Development

- Get code working in a friendlier environment prior to running on the target
  - Feedback
  - Efficient
- Decouple the application logic from the specific hardware dependencies
  - Bonus future benefit
- Feed events into the system
- Capture and verify the responses

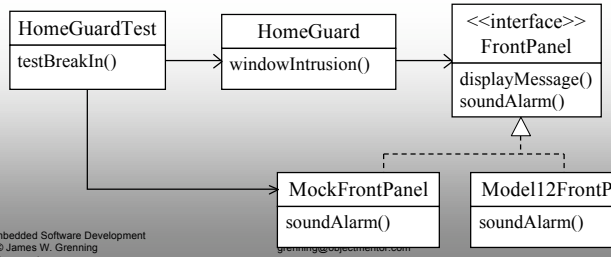
Design Impacts

- Test-driven design promotes testing a class in isolation
  - It must be decoupled from other classes
- Produces loosely coupled, highly cohesive systems
  - The hallmark of a good design
  - Object Oriented Design programming languages help
  - Encourages designs that adhere to Principles of Object Oriented Design (http://www.objectmentor.com/resources/articleIndex)
- A lot of the code is platform independent

Mock Objects

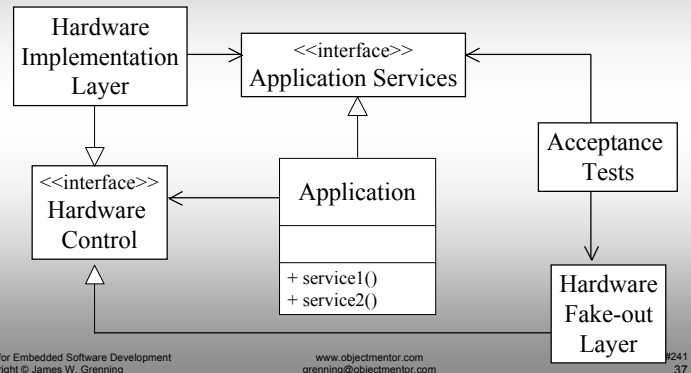


- Unwanted dependencies can be broken with an interface



Conceptual Architecture

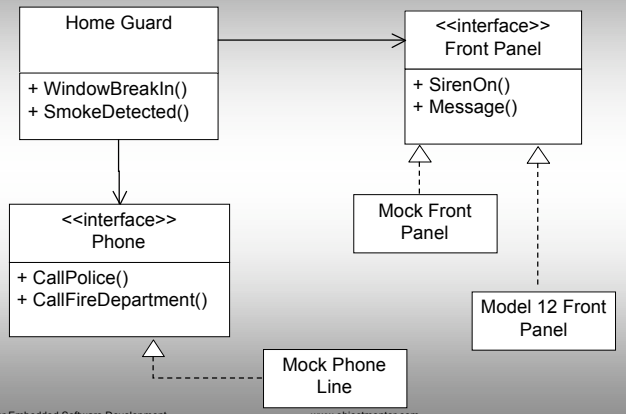
Tests use the application the same way the hardware does, only they bypass the hardware



Copyright © James W. Grenning
All Rights Reserved

Home Guard Architecture

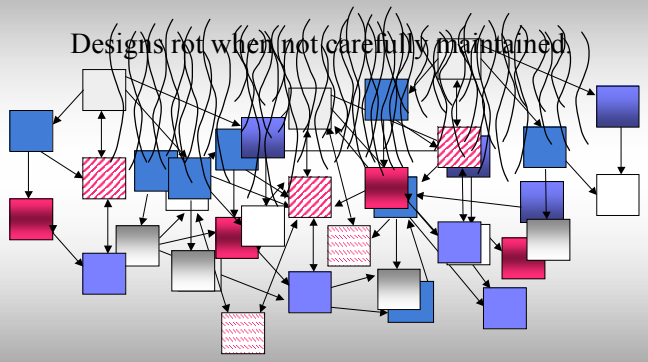
Notice the Separation of Concerns



Copyright © James W. Grenning
All Rights Reserved

What happens to a design over time?

Designs rot when not carefully maintained



Copyright © James W. Grenning
All Rights Reserved

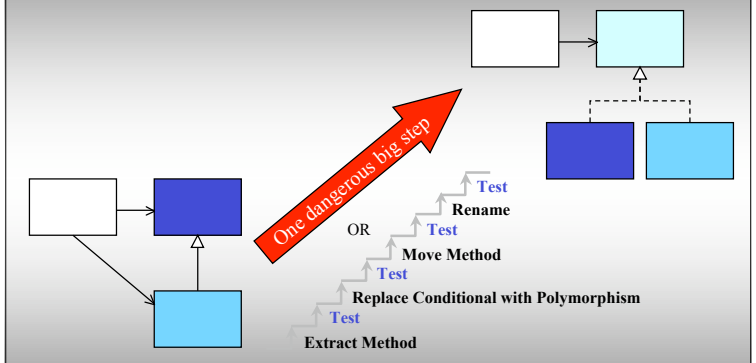
Neglect Is Contagious

- Disorder increases and software rots over time unless it is constantly repaired.
- Don't tolerate a broken window.

http://www.pragmaticprogrammer.com/ppbook/extracts/no_broken_windows.html

Copyright © James W. Grenning
All Rights Reserved

Transform Design with Small Steps



Tests are Documentation

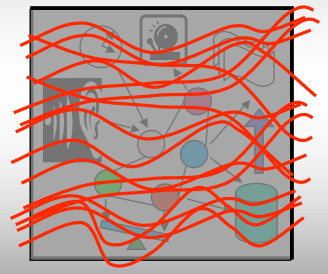
- Each test provides a working example
- Tests are superior to prose in that they are executable
- This documentation will tell you when it is out of sync with you code

Tests are a Simplification

Assuming a particular initial state

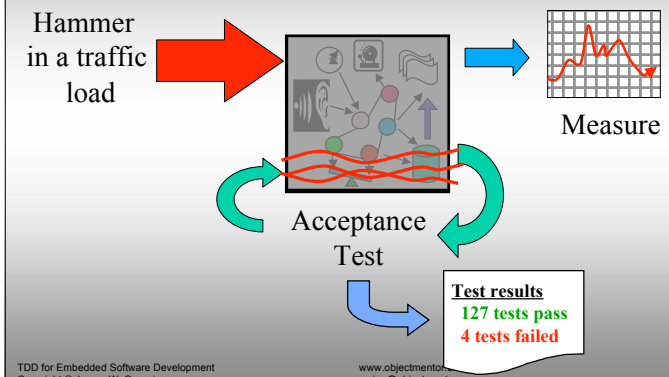
Stimulate the system

Check the response



Each test takes a path through the system

Hammer and Monitor (Load Test)



Tests Can be hard to Design

- Start early while the system is so simple it does not need tests
- Tests grow as the system does
- The effort is paid back many times over

The gift that keeps on giving

Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

But I don't have the right tools

- Inability to run code on the build system
  - Tests must be run on the target
- There is no unit test tool like JUnit or CppUnit for your development or execution environment
  - Write one or modify one
- Compiler incompatibility
  - Tests can be used to find compiler differences
- No OO programming language (Java, C++)

Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

No OO Programming Language

- OO promotes decoupling
- Decoupling enables testing

- C++ is C
- OO constructs in C++ can be taken advantage of for little or no additional cost
- If you are careful

- Decoupling can be done in C, but is harder and requires more discipline

Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

Review, Questions, Comments

- Iterative development reduces integration risks
- Automated tests provide a safety net while software evolves
- Designing for testability results in a improved cohesion and reduced coupling
- Testing is paid for by reduced defects and debugging sessions

Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

Productivity and Predictability

- Defects kill predictability:
  - Cost of fixing is not predictable
  - When they materialize is not predictable
- Test-driven is predictable:
  - Working at a steady pace
  - Results in fewer bugs
  - More productive than "*debug-later programming*"
- Test-driven programmers rarely need the debugger

Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

Learning Test Driven Design

- A skill which must be practiced
  - Initially awkward
- Requires discipline
  - Peer pressure
  - "I know how to write the class, but I don't know how to test it"
- *It's an addiction rather than discipline*
  - Kent Beck – Author of
    - Extreme Programming Explained
    - Test Driven Development

Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

To Learn More

- Read my papers or attend my classes
  - [ESC-349] Agile Embedded Software Development
  - [ESC-241] Test Driven Development for Embedded Software
  - [ESC-209] Object Oriented Design for Embedded Software Engineers
- Visit our Website
  - www.objectmentor.com
    - Training/Services
    - Resources/Articles

Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com

References

- James Grenning, various papers at http://www.objectmentor.com/resources/articleIndex
- Robert C. Martin, Agile Software Development: Principles, Patterns, and Practices, 2002
- Kent Beck, Test-Driven Development, 2003
- http://www.objectmentor.com
- http://groups.yahoo.com/group/testdrivendevelopment
- http://groups.yahoo.com/group/AgileEmbedded
- Test harnesses
  - [CPTEST] www.fitnesse.org/FitServers.CppFit.CppTestTools
  - [FITNESSE] www.fitnesse.org

Copyright © James W. Grenning
All Rights Reserved
grenning@objectmentor.com