Object Mentor, Inc
www.objectmentor.com

V1.2

Extreme Programming and Embedded Software Development

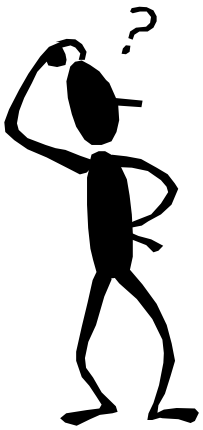
Class #462

James Grenning
Object Mentor, Inc

grenning@objectmentor.com

Source: Object Mentor Training

Project Management



What is the first thing known about a project?

James W Grenning
grenning@objectmentor.com

Source: Object Mentor Training

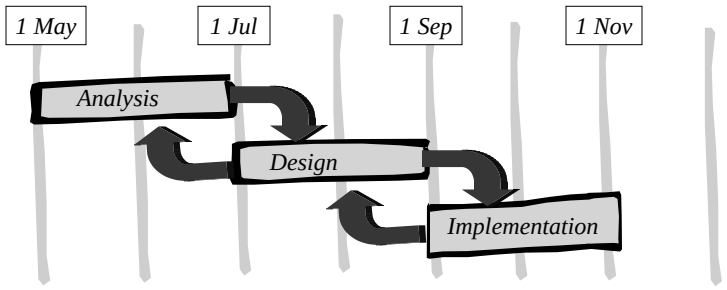
grenning@objectmentor.com

Problems Plaguing the Software Industry

grenning@objectmentor.com

Source: Object Mentor Training

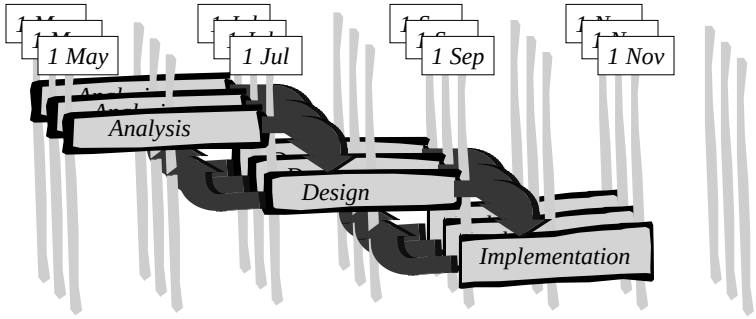
Simple Waterfall



grenning@objectmentor.com

Source: Object Mentor Training

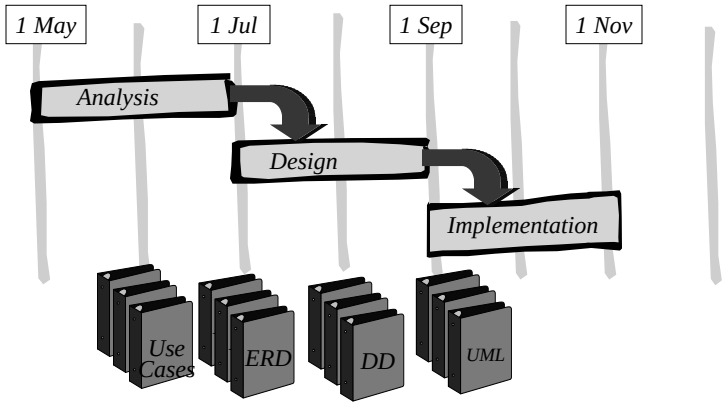
Waterfall as Royce Described



grenning@objectmentor.com

Source: Object Mentor Training

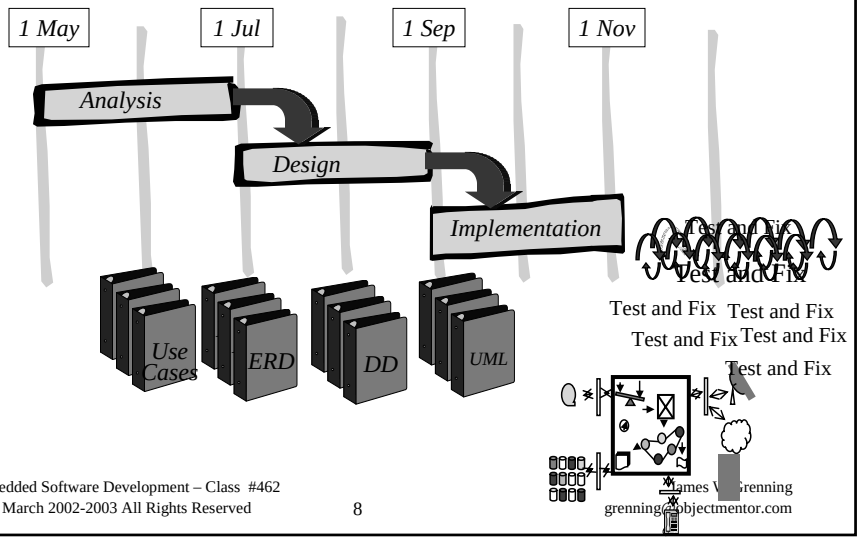
Waterfall as Practiced



James W Grenning
grenning@objectmentor.com

Source: Object Mentor Training

All the Problems are Saved for the End of the Project

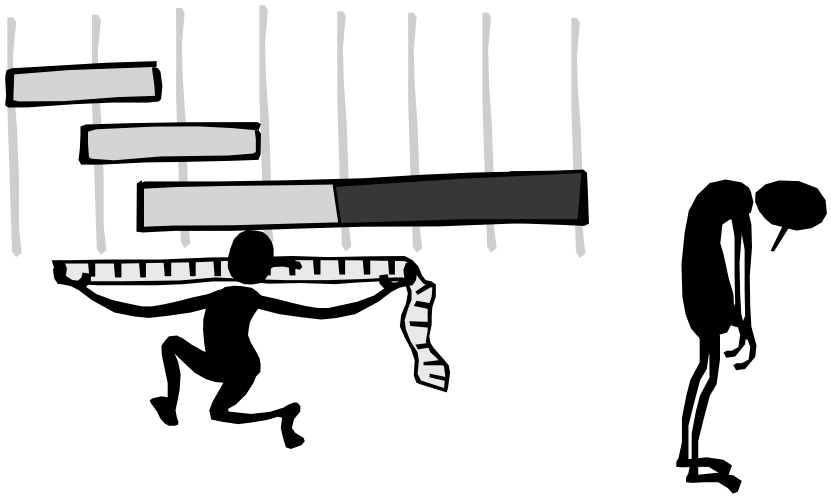


James W Grenning
grenning@objectmentor.com

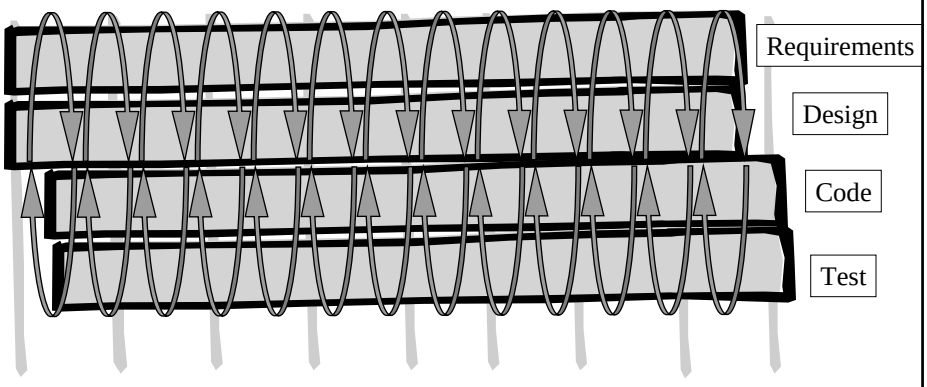
Poor Quality

Source: Object Mentor Training

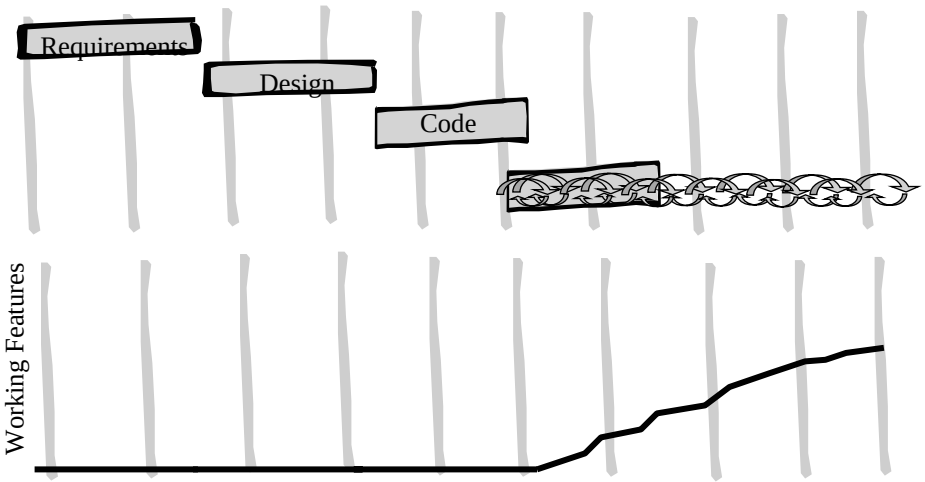
Unrealistic Expectations - Burn Out



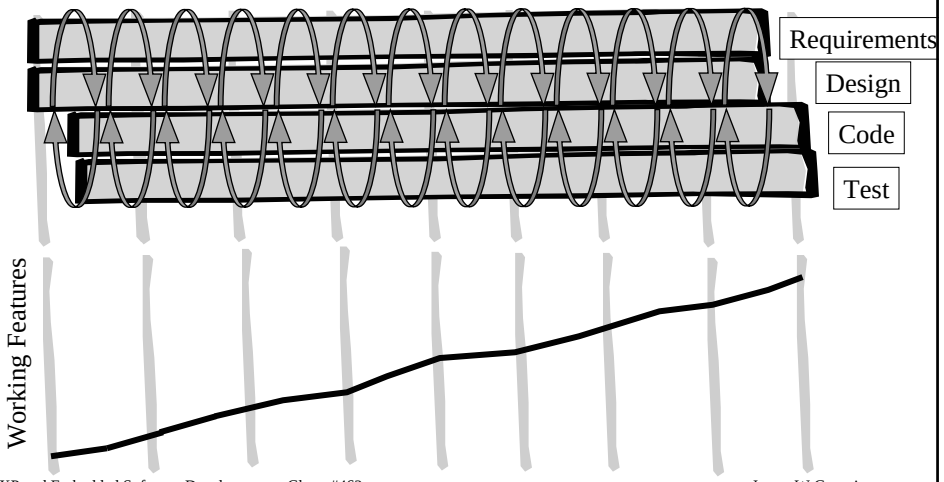
Iterative/Evolutionary Development Cycle



Typical development cycle – Working Features



Iterative/Evolutionary Development Cycle – Working Features

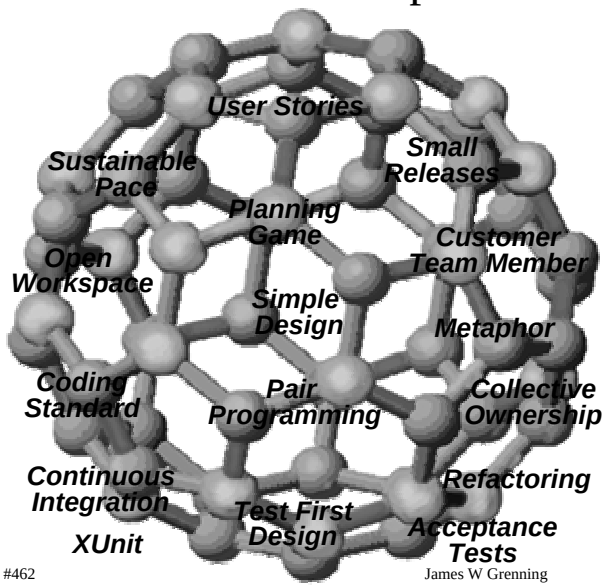


grenning@objectmentor.com

XP is a set of inter-related values and practices

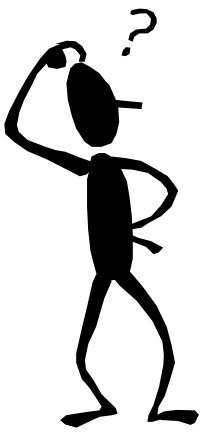
Values:

- Simplicity
- Communication
- Feedback
- Courage



grenning@objectmentor.com

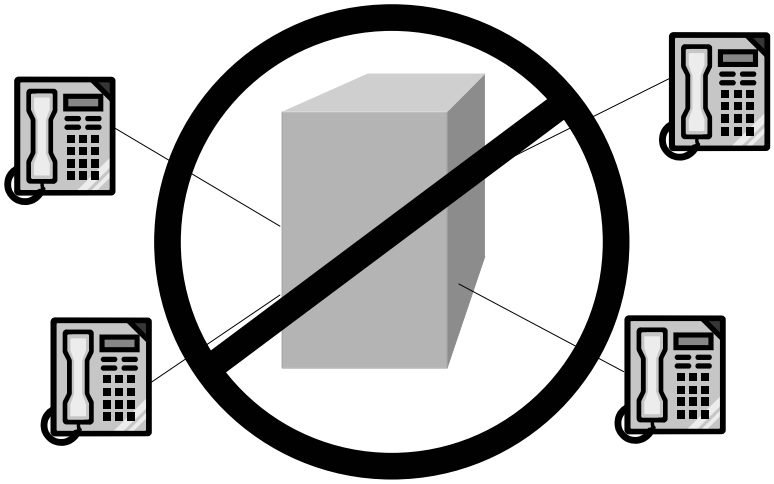
Embedded Software Development



What don't we have at the beginning of an embedded software project?

grenning@objectmentor.com

There's No Hardware



grenning@objectmentor.com

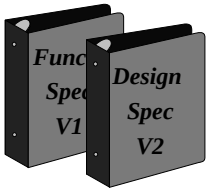
Embedded Software Development

- No Hardware until late in the project
- Target platform != development platform
- Computer is hidden from the user
- Limited IO
- Concurrent execution
- Resources are limited
- Real time

grenning@objectmentor.com

The Hardware is not Ready
How can we start the project?

- Write documents
- Have meetings
- Experiment
- Wait



Maybe there is another way

grenning@objectmentor.com

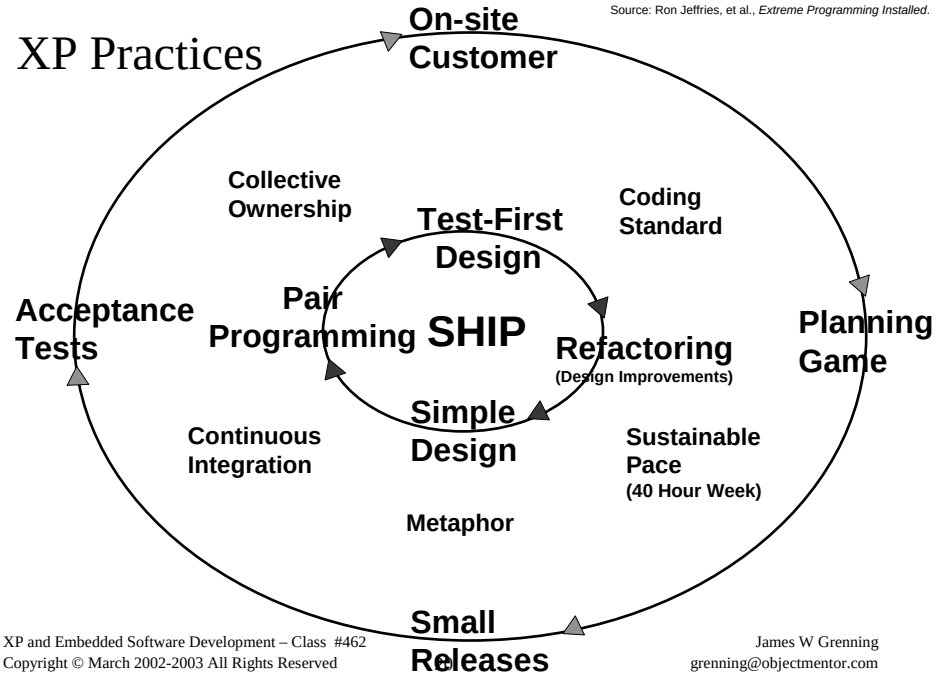
What is XP?

XP is an *Agile* methodology for developing software iteratively. XP encompasses a set of values, rights and best practices that support each other in incrementally developing software

- Values
  - Communication, Simplicity, Feedback, Courage
- Rights
  - Do quality work, have a life out of work
- Practices

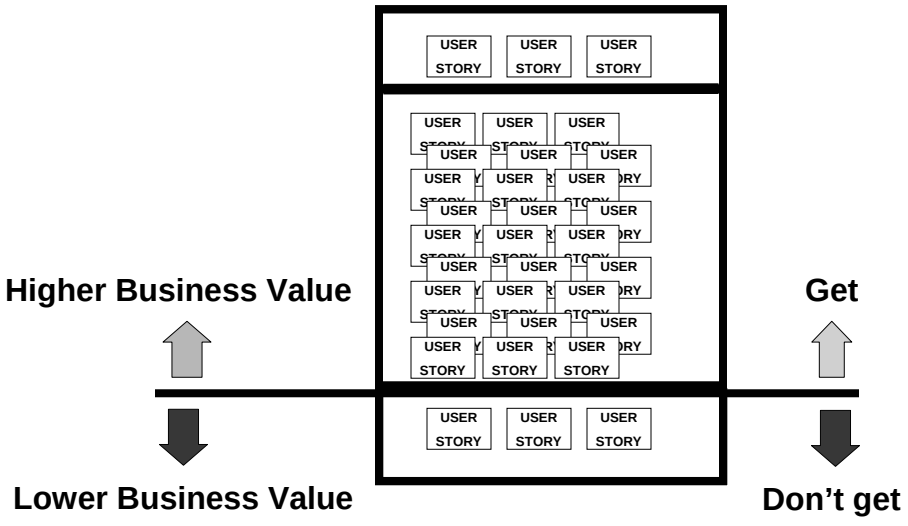
XP Practices

Source: Ron Jeffries, et al., *Extreme Programming Installed*.



Source: Object Mentor Training

Managing Scope



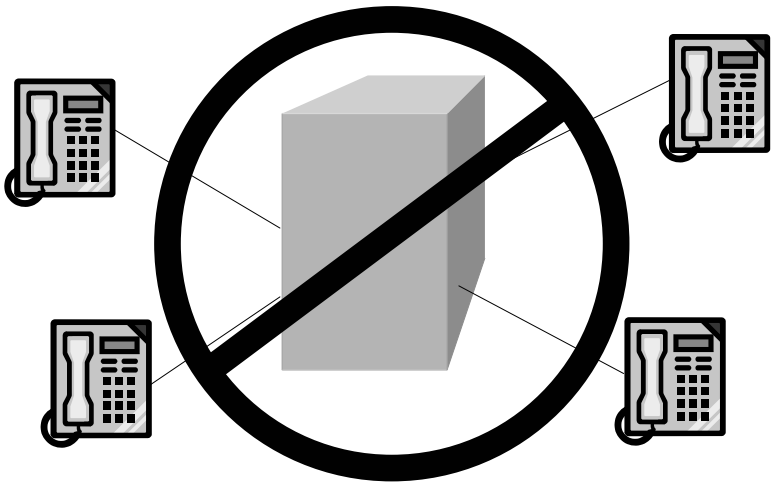
grenning@objectmentor.com

Stories

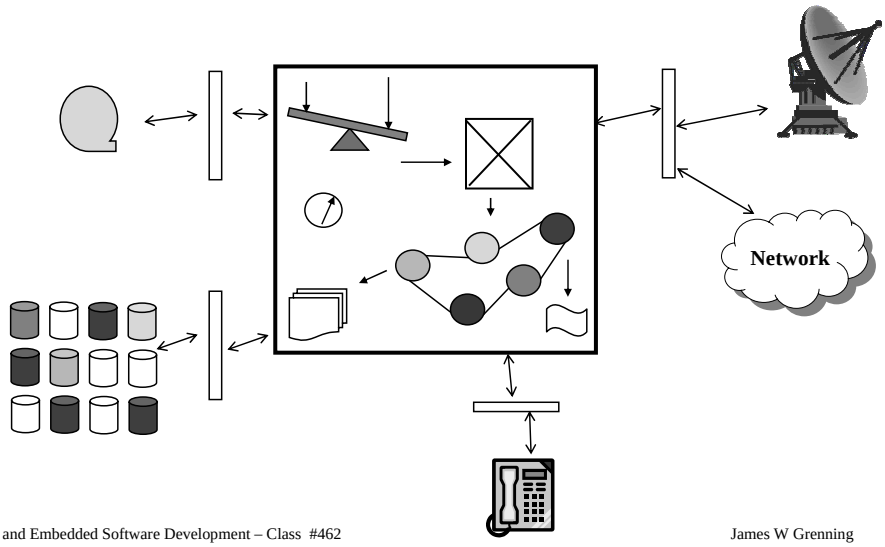
- Off-hook generates dial tone
- On-hook
- Extension calls extension
- Extension calls busy extension
- Flash, call transfer
- Flash, three way call
- Extension goes off-hook dials "9" and a line is available

James W Grenning
grenning@objectmentor.com

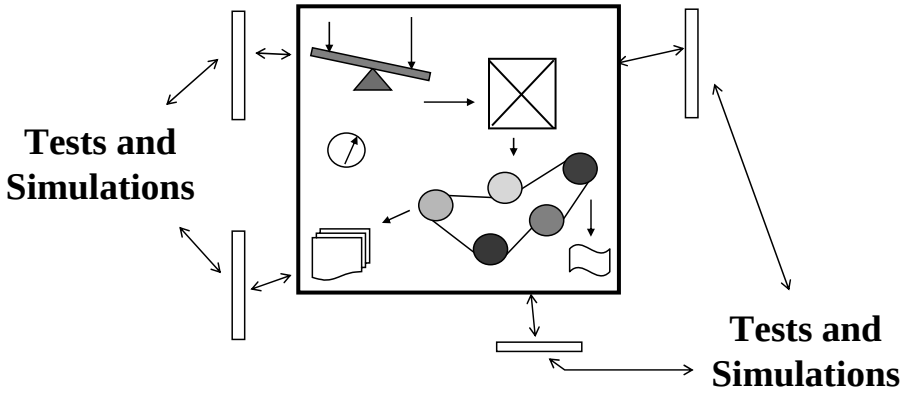
There's No Hardware



Separate Core System Logic from Hardware Specifics



Test Core System Logic Independent of Hardware Specifics



grenning@objectmentor.com

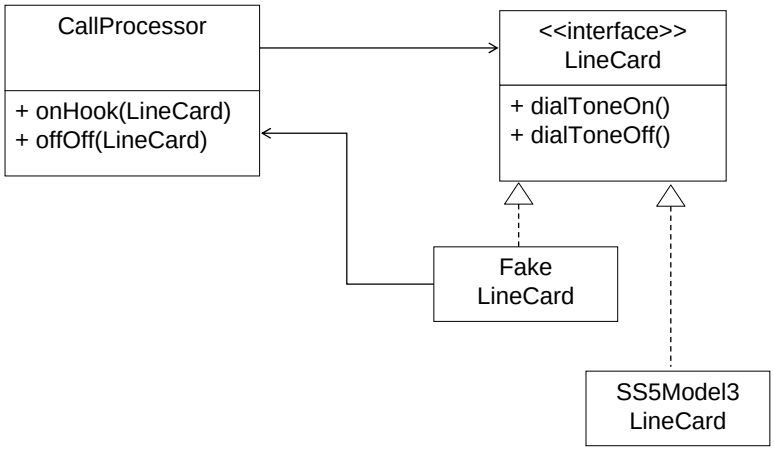
Acceptance Tests

- Slice off the hardware dependent outer layer
- Develop an application specific test language
- Run the full automated acceptance test suite multiple times per day, on your development platform

```
LINECARD 1 OFFHOOK
VERIFY LINECARD 1 DIALTONE
LINECARD 1 ONHOOK
VERIFY LINECARD 1 NODIALTONE
```


grenning@objectmentor.com

Initial Architectural Vision
(in UML)



grenning@objectmentor.com

Unit Test the Initial Architecture

```
//CallProcessorTest.cpp
#include "TestHarness.h"
#include "CallProcessor.h"
#include "FakeLineCard.h"

TEST(CallProcessorTest, DialTone)
{
    CallProcessor* cp = new CallProcessor();
    FakeLineCard* lc = new FakeLineCard();

    CHECK(lc->isDialToneOn() == false);
    cp->offHook(lc);
    CHECK(lc->isDialToneOn());

    delete cp;
    delete lc;
}
```


grenning@objectmentor.com

Separation of Concerns

```
//LineCard.h
class LineCard {
    public:
         virtual void dialToneOn() = 0;
         virtual void dialToneOff() = 0;
}
```

Separate the interface from the implementation

Keep the interface simple and easy to use

grenning@objectmentor.com

Separation of Concerns

Hide implementation details behind the interface

```
//FakeLineCard.h
#include "LineCard.h"

class FakeLineCard : public LineCard
{
  public:
    FakeLineCard()
    : dialToneOn(false) {
    }

    void dialToneOn() {
        dialToneOn = true;
    }
    void dialToneOff() {
        dialToneOn = false;
    }
    boolean isDialToneOn() {
        return dialToneOn;
    }
  private:
    bool dialToneOn;

}
```

Here is a hidden implementation for running tests

Separation of Concerns

Hide implementation details behind the interface

```
//SS5Model3LineCard.h
class SS5Model3LineCard : public LineCard
{
  public:
    SS5Model3LineCard () {
    }

    void dialToneOn() {
        //twiddle the bits to turn on dialtone
    }
    void dialToneOff() {
        //twiddle the bits to turn off dialtone
    }
  private:
    //Whatever member variables
    //are needed
    //etc.

}
```

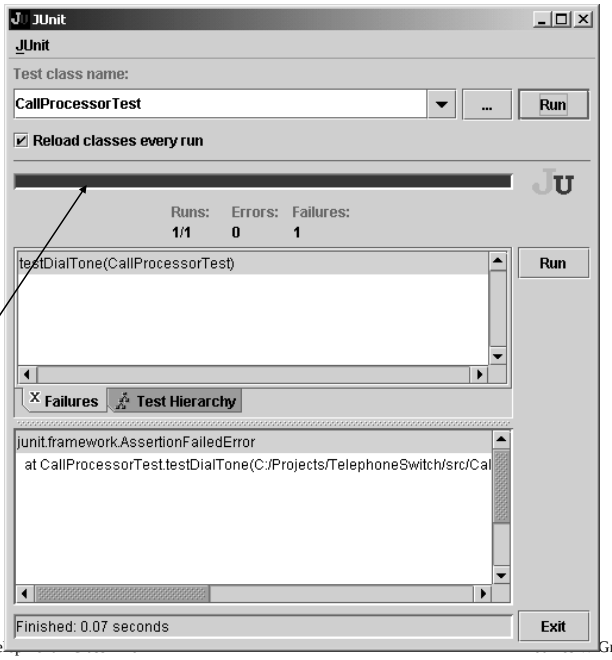
Here is a hidden implementation for For a specific line card

Initial Architecture Implementation

```
//CallProcessor.h
#include "LineCard.h"

class CallProcessor {
  public:

    void onHook(LineCard* lc) {
        lc->dialToneOn();

    }
    void offHook(LineCard* lc) {
        lc->dialToneOff();
    }
}
```

Unknown to CallProcessor, the test implementation is used

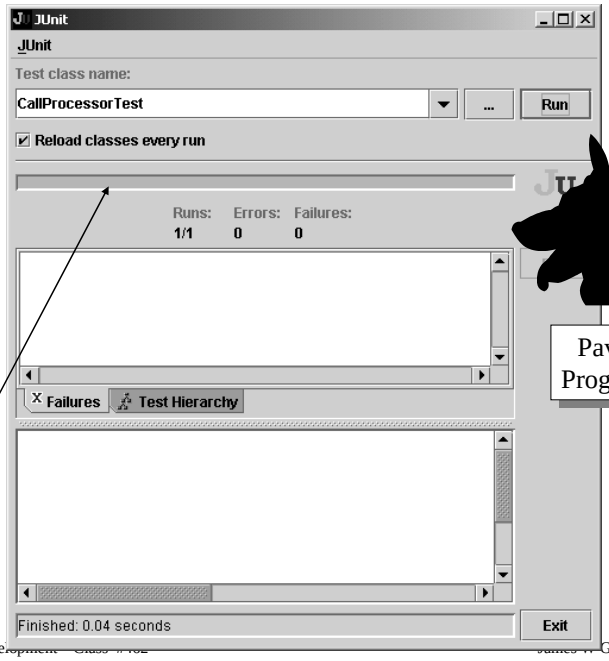
Run The Test



Red Bar means some test failed

Fix the bug

Run The Test

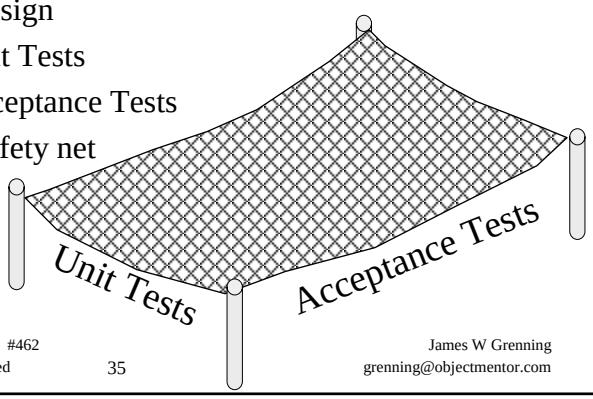


Pavlov's Programmer

Green Bar means all tests pass

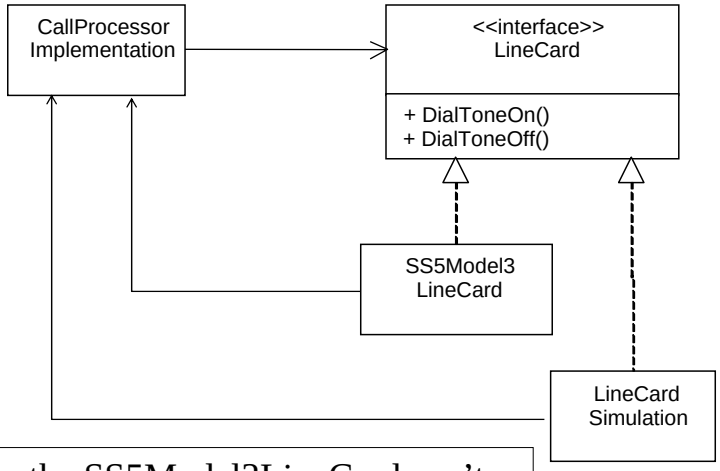
Evolutionary Design

- All Designs Evolve
- Refactoring (incremental design improvement)
- Automated testing supports design evolution
  - Test Driven Design
  - Automated Unit Tests
  - Automated Acceptance Tests
  - Tests are the safety net



James W Grenning
grenning@objectmentor.com

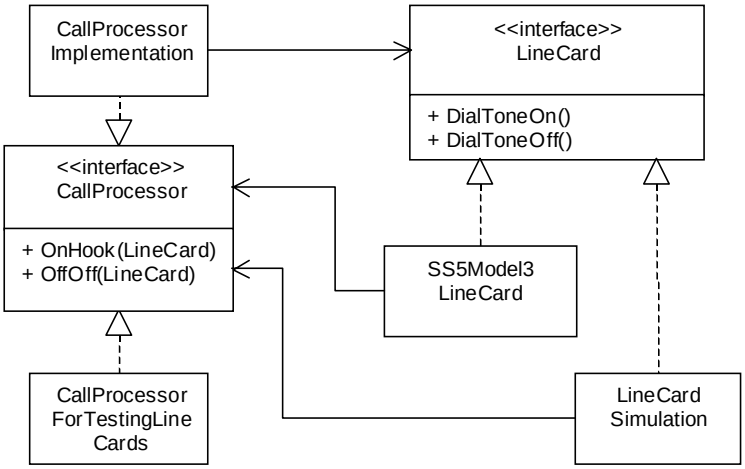
Evolutionary Design



Now the SS5Model3LineCard can't be tested w/o the CallProcessor

James W Grenning
grenning@objectmentor.com

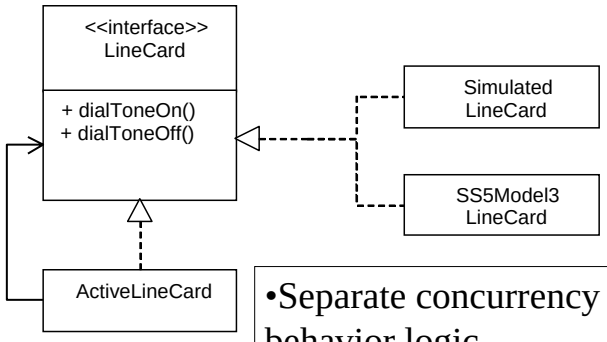
Evolutionary Design



grenning@objectmentor.com

Evolutionary Design

Concurrency - More separation of concerns



- Separate concurrency model from the behavior logic.
- ActiveLineCard applies the threading policy to the line card's operations

grenning@objectmentor.com

Limited Resources

- Simulation helps to get the logic right
- But, that may make you careless with resources
- Run in the target to understand all dependencies
- Measure critical resource usage
- Use data, not hunches to decide when to optimize
- Use data, not hunches to know when trouble is coming

grenning@objectmentor.com

Limited Resources

- Design tests that monitor critical resource usage
  - Test for memory leaks
  - Stack high water mark
- Use Big Visible Charts (BVC)
  - Memory usage
  - IO Bandwidth usage
  - Idle time
  - Hand draw or automatically generate
  - Look for troubling trends

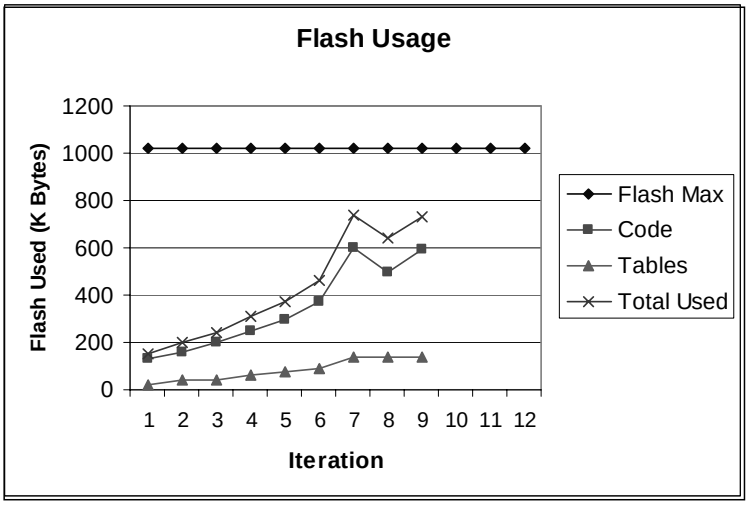
grenning@objectmentor.com

Memory Leak Test

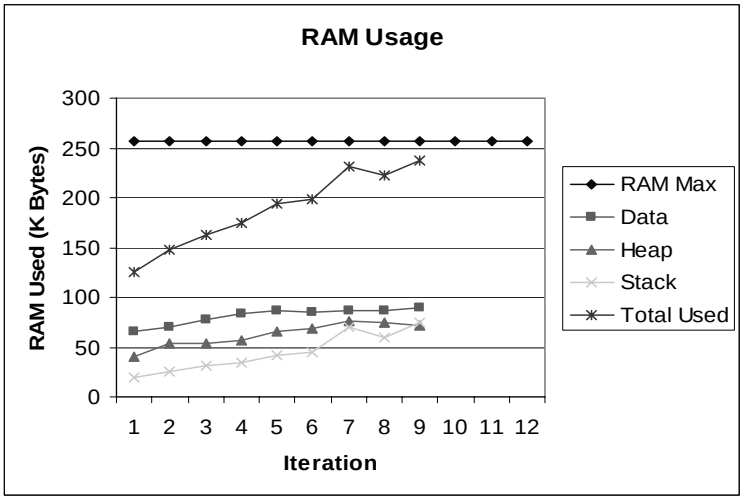
```
...
long heapSize = freeSpace();

//Run all unit tests

assertEquals(heapSize == freeSpace());
```

```
...
long int heapSize = freeSpace();

//The systems runs all acceptance tests
//The system exists

reportError(heapSize != freeSpace());
```


Memory Usage BVC

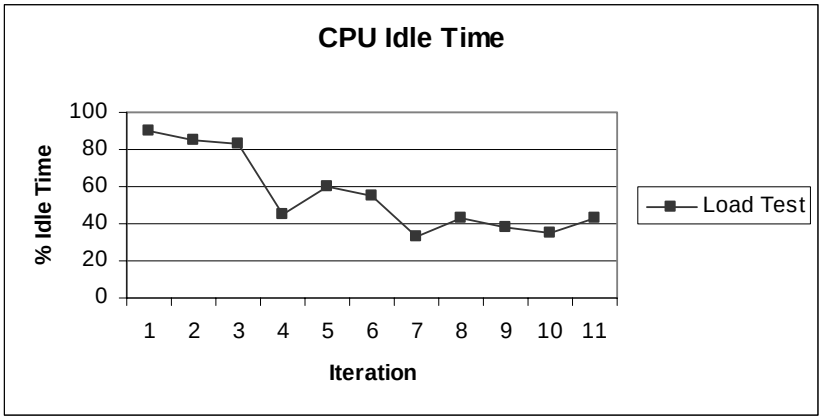


Memory Usage BVC



grenning@objectmentor.com

Idle Time BVC

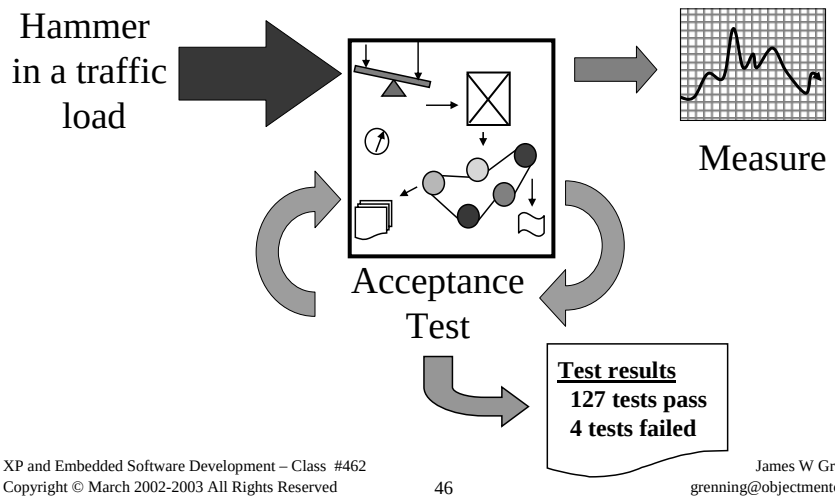


grenning@objectmentor.com

Real Time

- Performance Stories
  - 50,000 Lines
  - 10% of lines go off-hook simultaneously, dial-tone within 2 seconds
- Acceptance tests
- Unit tests with time check
- Hammer and Monitor
- Solve performance problems using data

Hammer and Monitor



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 a simple one
- Compiler incompatibility
  - Tests can be used to find compiler differences
- No OO programming language (Java, C++)

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

grenning@objectmentor.com

But I have a Safety Critical System

- Safety critical is about
  - Understanding the safety requirements
  - Documenting the process
  - Thoroughly testing the product
- Understanding the requirements
  - Every project regardless of process needs this for success.
  - Build teams around good people
- Thoroughly test the product
  - Aligns with XP testing goals
  - This is a function of good people, knowing the safety requirements

grenning@objectmentor.com

What about Documentation?

- Documentation is not evil. It is expensive and can get out of date.
- Challenge yourself: Understand why the documents are needed
- Are there other alternatives?
  - Can the documents be automatically generated
    - Can the traceability document be generated from the acceptance test running and recording the path trough the code?
  - Write the documents when they are the best way to meet the need
  - Can you write the documents as-built rather than as-anticipated?

grenning@objectmentor.com

Safety Critical

- XP is a incremental design and delivery strategy
  - This technique is orthogonal to the application domain
- Do you need to add more?
  - Safety requirements analysis
  - Additional reviews
  - Completeness testing
  - Safety critical systems design best practices
- More work needs to be done

grenning@objectmentor.com

Conclusions

- XP can help
  - Early progress prior to hardware
  - Testing safety net
  - Lower cost of change
  - Use BVCs for critical resources
  - Tools may be a problem
  - May have to explore new ground
    - Real time
    - Safety critical
- XP is an Agile development technique – look to the values for guidance

grenning@objectmentor.com

Agile Manefesto (www.agilemanifesto.org)

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

grenning@objectmentor.com