Improving Testing's Value through Customer Collaboration

Gerard Meszaros

StarEast2008@gerardm.com

Instructor Biography

Gerard Meszaros is a Calgary-based consultant specializing is agile development processes. Gerard built his first unit testing framework in 1996 and has been doing agile test-driven development ever since. He is an expert in test automation patterns, agile project management and improving usability practices on agile projects.

Gerard has applied automated unit and acceptance testing on projects ranging from full-on eXtreme Programming to traditional waterfall development and technologies ranging from Java, Smalltalk and Ruby to PLSQL stored procedures and SAP's ABAP.

He is the author of the Jolt Productivity Award winning book xUnit Test Patterns – Refactoring Test Code and a frequent presenter on agile development at major conferences and user groups.

Gerard Meszaros
StarEast2008@gerardm.com

Objectives of Tutorial

- **Understand the value of collaboration**
  - Why collaboration is always better than traditional competative or confrontational interactions
- **Understand the opportunities for collaboration**
  - How can testers add more value throughout the software development lifecycle. What activities and deliverables can they add value to?
  - How to keep your job from being off-shored!
- **Understand the techniques for collaboration**
  - How do we get people to collaborate?
  - Leadership vs. Management

Teaching Method

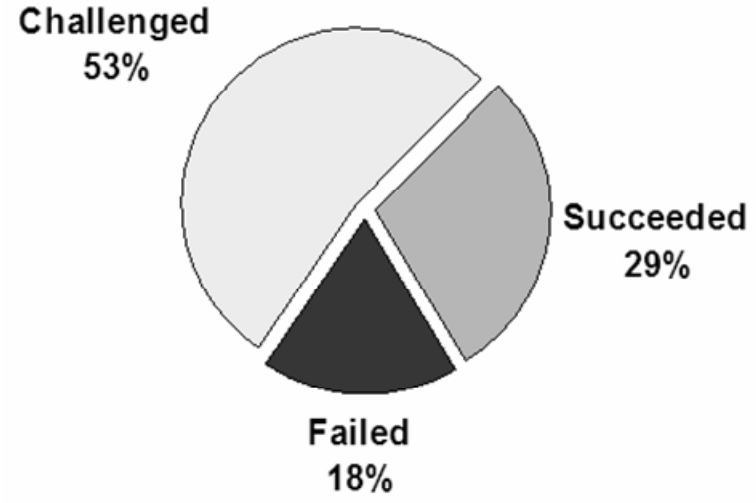
Most adults learn best by doing

1. **Introduction to topic**
   - PowerPoint presentation describing concepts.
   - Sample code or other concrete examples
2. **Short Exercise**
   - Work in small groups trying out concepts
   - About 5-15 minutes
3. **Short Discussion**
   - Someone from each group provides one answer to one question
   - Round-robin so every group is heard (eventually)

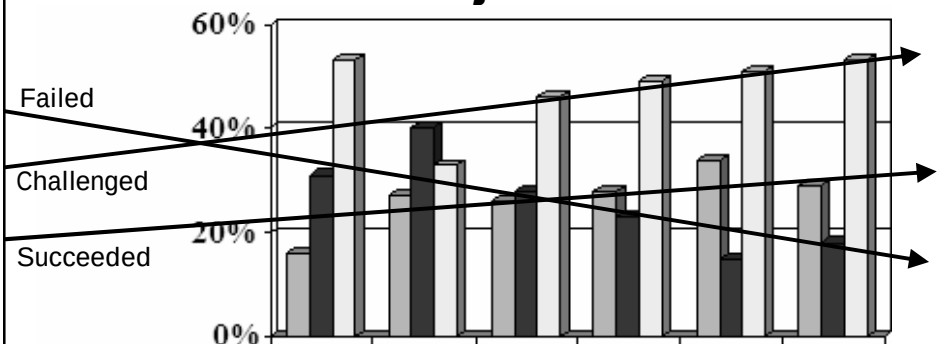
Agenda

- **The State of the Union**
  - How are we doing as an industry or profession?
- **S/W Development Value Stream**
- **How Can Testers Help Development?**
- **How can Testers help Product Management?**
  - Define product concept
  - Validate product design
  - Clarify requirements
- **Collaboration Techniques**
- **Summary & Closing Comments**

CHAOS 2004 – Resolution of Projects

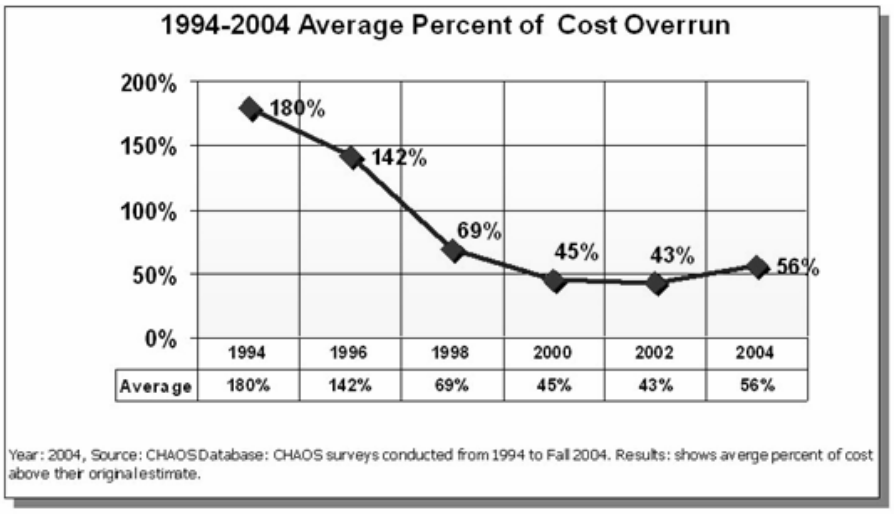


CHAOS 2004 – Project Success Trend



	1994	1996	1998	2000	2002	2004
Succeeded	16%	27%	26%	28%	34%	29%
Failed	31%	40%	28%	23%	15%	18%
Challenged	53%	33%	46%	49%	51%	53%

CHAOS 2004 – Average % Cost Overrun



Is This Good Enough?

- In a word:
  - NO!
- Is This the Best We Can Do?
  - NO!

CHAOS 2004 – Top 10 Reasons for Success

- Collaboration
- Communication
- Collaboration
- Highly Collaborative
- via Collaboration
- Agile is very formal

* Excerpts from Aug 2006 CHOAS Interview

InfoQ: Your work on project failure is really a search for "how to succeed", isn't it? I noticed your list of Project Success Factors includes ***#5: Agile Development***. As in, Agile Software Development?

Jim Johnson (CEO Standish Group): Oh, absolutely! **I'm a big believer in Agile**, having introduced the iterative process in the early 90s and followed with the CHAOS reports on quick deliverables. We're a real flag waver for small projects, small teams, Agile process. ..

Gordon Divit (VP Acxsys Corp): **Agile** has helped by chunking projects into small pieces. Even if you start to get it wrong, you know it early on. Not like the old **waterfall** projects where you go into a cave and find out the **bad news two years later**...

Jim Johnson: I think that's the secret - incremental. **I think that's why we're seeing improvement.**

See: http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS

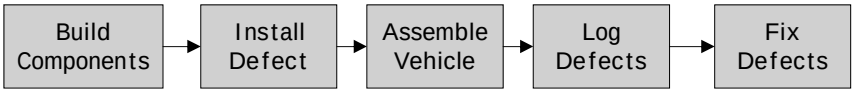
Agenda

- **The State of the Union**
- **S/W Development Value Stream**
  - What's involved in delivering value thru software?
- **How Can Testers Help Development?**
- **How can Testers help Product Management?**
  - Define product concept
  - Validate product design
  - Clarify requirements
- **Collaboration Techniques**
- **Summary & Closing Comments**

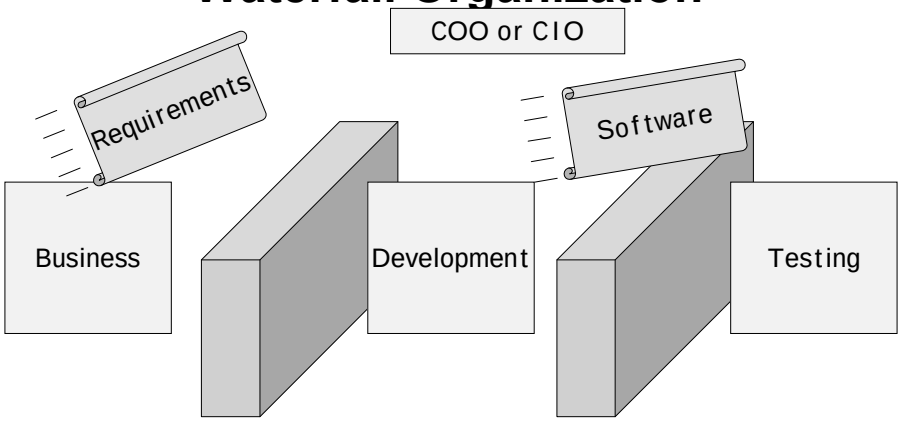
Value Stream Mapping

- A way to understand all the work required to produce a product or deliver a service
- Often used to identify waste in the delivery process in a systematic way
- Example: "Waterfall" Software Development

- Example: American Car Assembly Line



Waterfall Organization



- Different bosses & measures of success
- Leads to conflicting behavior

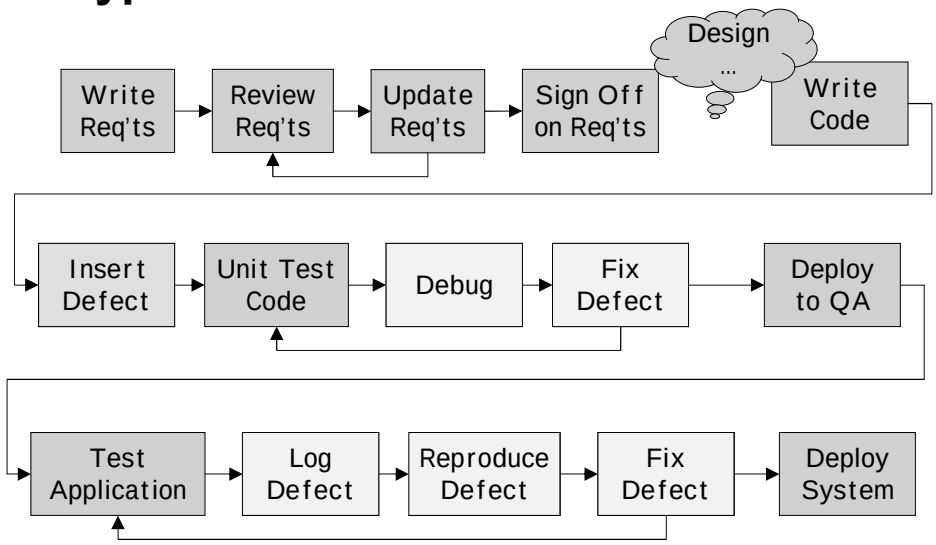
The Problem with "Over the Wall" Handoffs

- **Testing is done too late in the project to influence the delivered product significantly**
  - Missed opportunity to take advantage of testing's skills and experiences
- **Discovery of defects causes rework of software resulting in long test&fix cycles.**
  - This results in longer "concept to cash" timeframes.
- **Positioning of testing as the guardians of quality:**
  - Sets up conflict with the development organization
  - Can result in development abdicating any responsibility for quality resulting in more bugs and rework

Danger of "Over the Wall"

- **If communication is arms-length, what prevents it from being off-shored where:**
  - Costs are lower,
  - Can do overnight testing ("around the clock development") reducing project timelines
- **Beyond running tests, what value can testers (& test organizations) provide?**

* Typical "Waterfall" S/W Value Stream



Exercise

Working by yourself, please write down in bullet form on a piece of paper your answers to the question that follows.

Please use short phrases. You have 2 minutes to answer:

"What are the skills & experiences of testers that can add value to (the non-testing phases) of a project?"

Exercise

- **In small groups**
  - Start a new **merged** list
  - (Please keep your original lists for the next exercise.)
- **Take turns reading an item from your list**
  - Add it to the merged list UNLESS the item is similar to another item already on the list.
  - If this idea triggers any other ideas, add them to the list also.

You have 2 minutes to Merge your lists.

GO!

Exercise

- **I need 2 volunteers to act as scribes.**
  - You'll be recording alternate items on flip charts
- **Now let's go around the room and have each group add a new, distinct item to the list. We'll start from my left and proceed clockwise around the room. If you don't have any items that were not already mentioned then just say "Pass".**
- **So, let's hear your collection of:**

"Skills & experiences of testers that can add value to (the non-testing phases) of a project?"

Debrief

Refer back to your original piece of paper. How many items did had you written down?

Compare this list to the list you made with your small group. Would you characterize the merged list as:

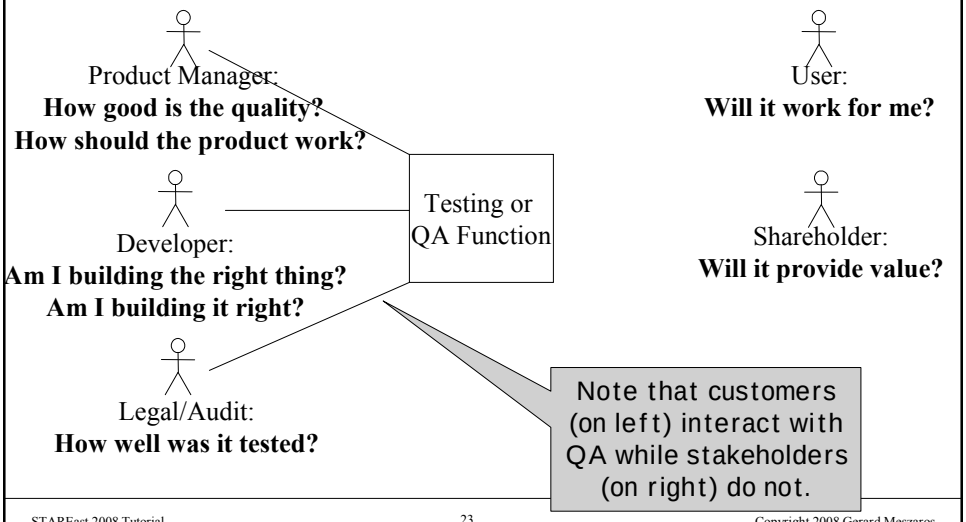
1. Much better than the original list
2. Somewhat better than the original list
3. Equivalent to the original list
4. Somewhat weaker than the original list
5. Much less useful than the original list

Repeat comparing your original list with the list we made as a large group

Who are Testing's Customers?

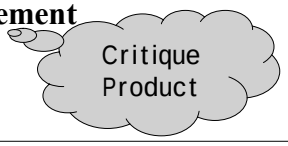
- **and what service does Testing provide to them?**

Testing as a Service:
Stakeholders vs. Customers



Purpose of Tests

	Support Development	Support Product Management
Business Facing	Acceptance Tests Regression Tests	Usability Tests Exploratory Tests
Technology Facing	Unit Tests Component Tests	Property Tests (Response Time, Security, Scalability)



How Tests Support Product Management

- **Help Product Management understand quality**
  - Provide a "Scorecard" for the Product
  - Whenever they need it
    - » **Not just at Release Milestones**
    - » **Scorecard can improve visibility into status of development**
- **Help Product Management understand business impact of defects**
  - Risk = Probability * Consequence
  - Consequence must be expressed in business terms
    - » **Direct Support costs**
    - » **Lost opportunity cost (of people diverted to support)**
    - » **Cost of damage to reputation**

How Testing Could Support Product Mgnt (2)

- **Help Prod Management define requirements**
  - Provide input during Product definition
    - » **Testers often understand user needs better than developers or Business Analysts**
    - » **Or they provide complementary insight**
- **Help Prod Management validate requirements**
  - Test early where cost of change is low,
  - Before software is developed and unit tested

How Tests Support S/W Development

- **Before code is written**
  - Tests as Specification
    - » **What the code ***should*** do**
    - » **Clarification of the Requirements**
- **After code is written**
  - Tests as Documentation
    - » **What the code ***does***
  - Tests as Safety Net
    - » **Provide Rapid feedback on quality (A.K.A. Bug Repellent)**
  - Defect Localization
    - » **Reduce cost to fix by Minimizing Debugging**

Agenda

- **The State of the Union**
- **S/W Development Value Stream**
- **How Can Testers Help Development?**
  - Sources of Wasted Time & Effort
  - Avoiding Defects thru Better Understood Requirements
  - Providing Feedback Earlier
- **How can Testers help Product Management?**
- **Collaboration Techniques**
- **Summary & Closing Comments**

Quality Inspiration: Lean Manufacturing

- **The manufacturing process that has made Toyota the quality leader amongst major automobile manufacturers.**
- **Also known as Just-In-Time manufacturing**
- **Basis for Toyota's Lean Product Development process**
- **Being adopted by Big 3 Automakers in N.A.**

The same principles can be applied to software …
… with similar impact on quality

7 Key Principles of Lean Software

1. **Eliminate Waste**
2. **Build Quality In**
3. **Create Knowledge**
4. **Defer Commitment**
5. **Deliver Fast**
6. **Respect People**
7. **Optimize the Whole**

These all support the core concept:

- Deliver Value

The 7 Wastes of Lean Manufacturing

- In-Process Inventory
- Over-Production
- Extra Processing
- Transportation
- Motion
- Waiting
- Defects

The 7 Wastes of Lean Software

• In-Process Inventory	• Partially-Done Work
• Over-Production	• Extra Features
• Extra Processing	• Relearning
• Transportation	• Handoffs
• Motion	• Task Switching
• Waiting	• Delays / Queueing
• Defects	• Defects

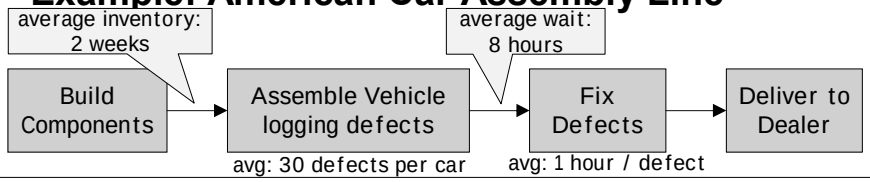
e.g.: Untested Software

or Unnecessary Process

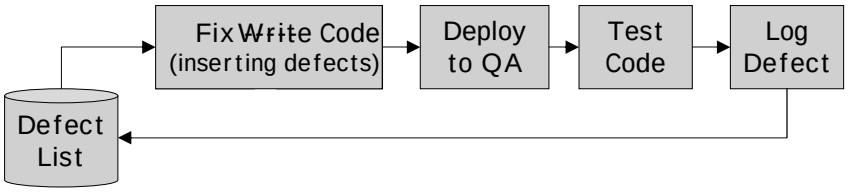
From: Implementing Lean Software Development
by Mary & Tom Poppendieck

Value Stream Mapping

- **A way to understand all the work required to produce a product or deliver a service**
- **Often used to identify waste in the delivery process in a systematic way**
- **Can calculate metrics such as:**
  - Average cycle time
  - Process cycle efficiency
- **Example: American Car Assembly Line**



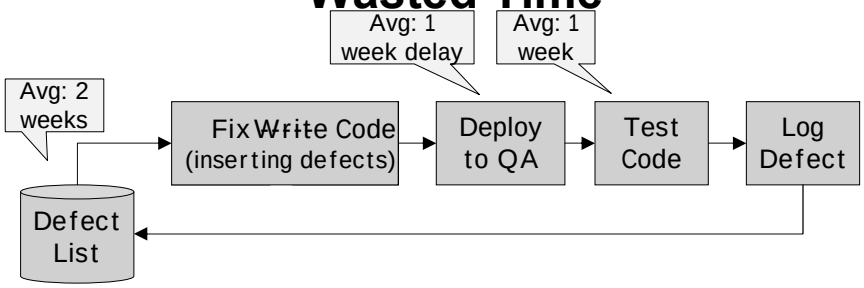
S/W Development Value Stream



What is this costing us?

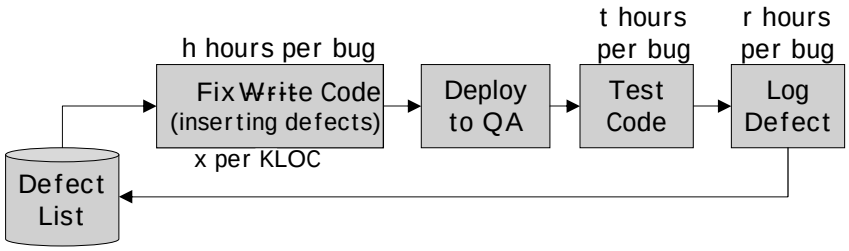
- **In delayed delivery?**
- **In extra effort?**

Wasted Time



- **Imagine it takes 2 test cycle to fix enough bugs to release**
  - 2 weeks to fix bugs
  - 1 week waiting to deploy
  - 1 week retesting
- **Delivery is delayed by 8 weeks!**
- **How much is an 8 week delay worth to your company?**

Wasted Effort



- **Imagine we have 100 KLOCS of code delivered with a defect rate of 10 bugs per KLOC.**
  - 1000 bugs; 1 hour each to report;
  - 8 hours each to fix; 3 hours each to test
  - 12,000 hours of wasted effort
  - = 1500 person days or 7.5 person years = ??? $$

Exercise:

- In small groups, brainstorm:
- "How could Testing help Development avoid inserting defects?"
- "What would get in our way?"
- "How could we surmount the obstacles?"
- Spend at least 3 minutes on each

Exercise Worksheet: Ways Test could Help Dev't

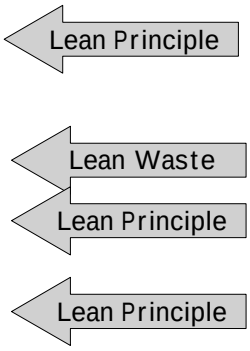
Exercise Worksheet: Potential Obstacles

Exercise Worksheet: Ways to Surmount Obstacles

Debrief

Ways to Help Development

- **Communicate requirements more clearly**
  - By sharing "what done looks like" before development starts
  - Lets development "build quality in"
- **Verify conformance earlier**
  - By testing early & often
  - Reduces inventory of untested software
  - Allows faster delivery
- **Help development unit test better**
  - Help them build quality in
  - By sharing testing skills with them



Potential Obstacles

- **It takes too much effort to test early & often**
  - Automation of tests can reduce effort significantly
  - Picking the right test automation strategy is key
- **Testing doesn't have any spare cycles because it is too busy testing the last release or another product**
  - Need to get out of this Catch-22 somehow
  - Maybe hire some contract testers to free up some cycles for new initiatives

Agenda

- **The State of the Union**
- **S/W Development Value Stream**
- **How Can Testers Help Development?**
  - Sources of Wasted Time & Effort
  - Avoiding Defects thru Better Understood Requirements
  - Providing Feedback Earlier
- **How can Testers help Product Management?**
- **Summary & Closing Comments**

How Can Testers Help Development?

- **Tell them how you will test their software**
- **Before they build it!!**

Why Provide Tests/Examples Earlier?

- **Show what "done looks like"**
  - Stephen Covey's 2nd "Habit": "**Begin with the End In Mind"**
- **Enumerate all the capabilities that must be supported**
  - Identify the success paths
  - Identify all the failure & error scenarios
- **Provide detailed examples of input data and responses**

"*The Seven Habits of Highly Effective People*"
by Stephen R. Covey published by Free Press

Why Not?

- **"If they know how we'll test it they'll write their code to pass our tests!"**
- **But isn't that the whole point?**

Unless,

- **You want lots of bugs**
  - so that you can find them
  - so that you can fix them

Rewarding Finding/Fixing Bugs

See cartoon at:
- **http://i18.tinypic.com/34461wi.jpg**

or:
- **http://www.flubu.com/comics/dilbert2.gif**

Or, search Google Images for
'dilbert "write me a minivan"'

- **Is this really what we want?**
- **What's the alternative?**

Rewarding Quality

- **Collaboration won't happen if individualism is what is rewarded.**
  - Whole Team needs to sink/swim together
- **Reward system needs to reinforce this**
  - e.g Team Performance Ranking
- **Quality measured in the field,**
  - Not lack-of-quality measured in test lab

Lean Principle: Optimize the Whole

Examples vs. Tests

Examples:

- **Show what "done looks like"**
- **Enumerate all the capabilities that must be supported**
- **Provide detailed examples of input data and responses**

Support Development

Tests:

- **Try to find defects in built software**
- **Verify all the capabilities that should now exist in the software**
- **Exercise system with a variety of inputs**

Critique the Product

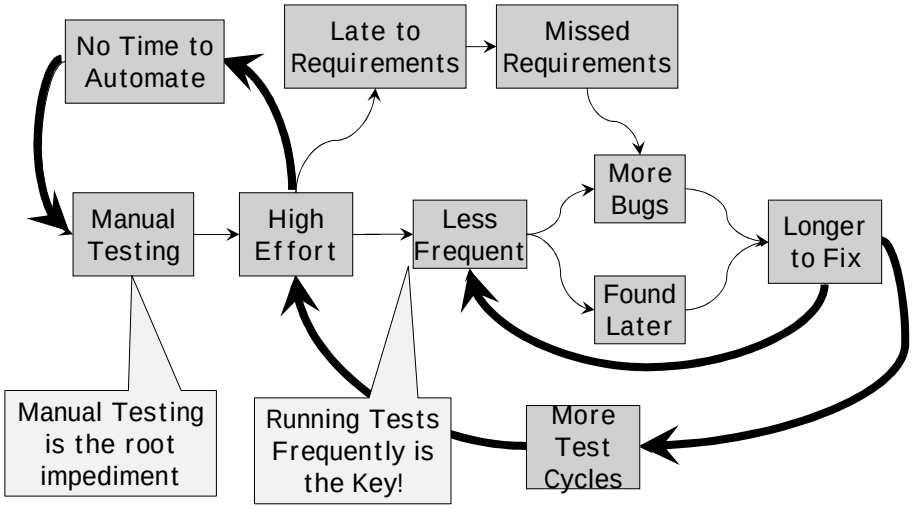
Agenda

- **The State of the Union**
- **S/W Development Value Stream**
- **How Can Testers Help Development?**
  - Sources of Wasted Time & Effort
  - Avoiding Defects thru Better Understood Requirements
  - Providing Feedback Earlier
- **How can Testers help Product Management?**
- **Summary & Closing Comments**

Getting Feedback Sooner

- **The sooner development gets feedback, the less effort gets wasted:**
  - Building more software based on wrong assumptions
  - Trying to remember how the software works
    - » **Because it's been weeks or months since I wrote it**
  - Debugging the software
    - » **Because I can't remember what/when I changed it to work that way**

Testing & the Cost of Change



The Problem with Manual Testing

- **Not very repeatable**
  - Either cannot remember exactly how it was last tested,
  - Or, too expensive to document detailed procedures
- **Very effort-intensive**
  - How long will it take to rerun all the tests?
  - How often will you actually do it?
- **Doesn't solve the communication problem**
  - Tests typically prepared and executed once application is built

Why Automate Tests?

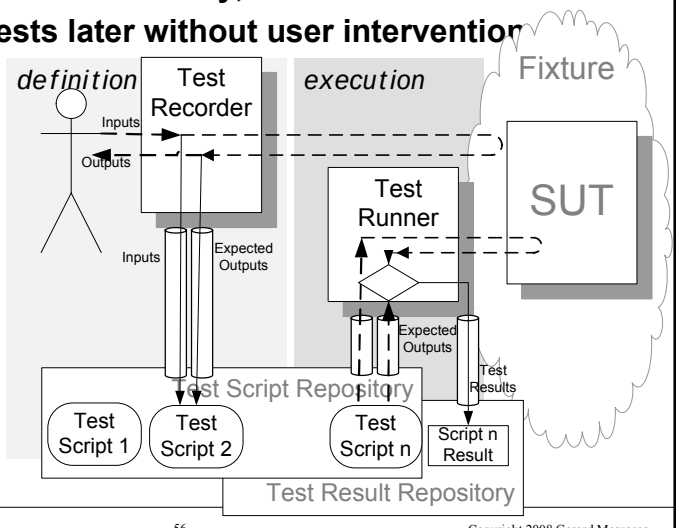
Make measuring quality:

- **Less expensive**
  - Fewer resources can run tests more rapidly
- **More timely**
  - Can run tests more often; whenever it would help development

Recorded Tests

- **User executes tests manually; tool records as tests**
- **Tool replays tests later without user intervention**

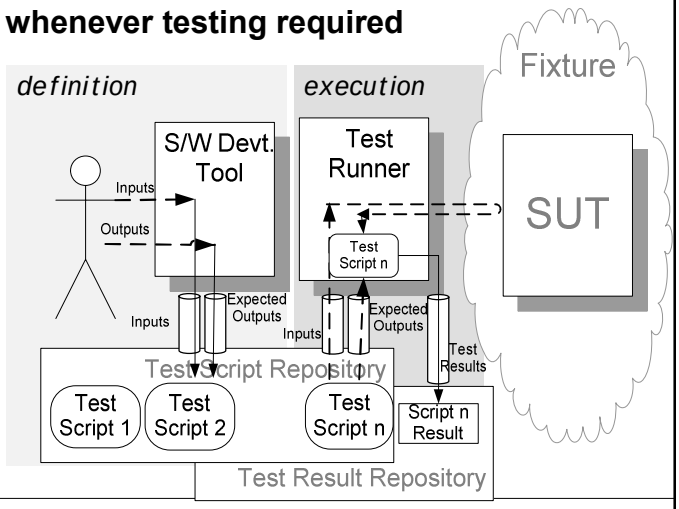
The tests are data interpreted by the test tool.



Scripted Tests

- **Tester writes test code to exercise the software**
- **Test code run whenever testing required**

The tests are a program that tests the program



Which Automation Strategy?

Recorded Tests:

- **Fast to create**
  - "No coding skills Required"
- **Unmaintainable**
  - Coding skills required
- **Don't support development process**
  - Only useful after the fact
- **Done by Testing**

Scripted Tests:

- **Harder to create**
  - Coding skills required
- **Can be done ahead of time**
  - Supports the development process
- **Requires design for testability**
- **Done by techies**
  - Development
  - Technical testers

Neither is Optimal for "Unjamming"

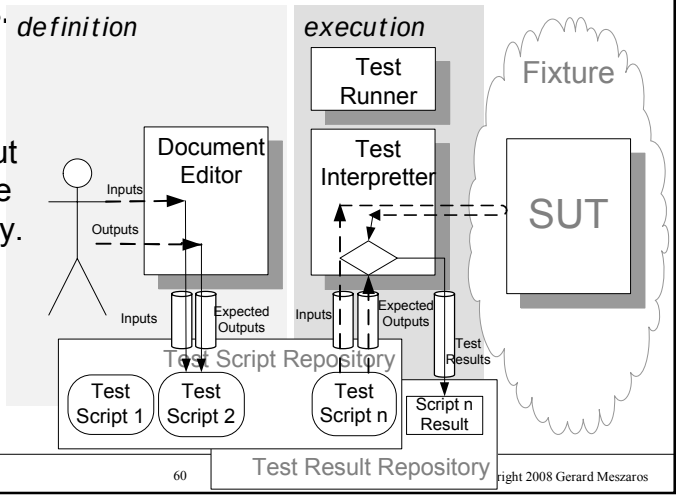
Collaborative Test Automation

- **Need an approach that divides work into two categories**
- **Describing the tests**
  - Non-technical
- **Interfacing the tests with the SUT**
  - Technical

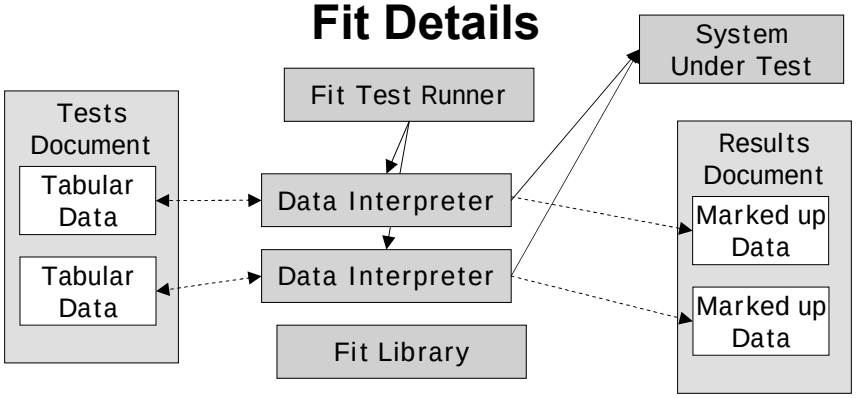
Data-Driven Testing using Fit

- **The tests are expressed as tabular data by users.**
- **The tests are read & executed by a test interpreter written by developers.**

Prepared like Scripted Tests but with a much more limited vocabulary.



Fit Details



- **Fit provides a library of Data Interpreters that can be customized to translate Tabular Data into interactions with the SUT**
- **Fit marks up Tabular Data with test results**

Fit Example: Column Fixture Test

PayrolFixtures.WeeklyCompensation			
Standard Hours	**Holiday Hours**	**Hourly Wage**	**Pay( )**
40	0	10	$400
40	0	20	$800
41	0	20	$830
40	1	20	$840
41	1	20	$870

Fit Example: Column Fixture Test

PayrolFixtures.WeeklyCompensation			
Standard Hours	**Holiday Hours**	**Hourly Wage**	**Pay( )**
40	0	10	$400
40	0	20	$800
41	0	20	$830
40	1	20	$840 *expected* $800 *actual*
41	1	20	$870 *expected* $830 *actual*

Note: Fixtures can have side effects. Pay() could add these rows into a database table

Walkthrough: Column Fixture Test

```
public class WeeklyCompensation ColumnFixture {

   public int StandardHours;

   public int HolidayHours;

   public Currency Wage;

   public Currency Pay( ) {
      WeeklyTimesheet timesheet = new
         WeeklyTimesheet( StandardHours, HolidayHours);
      return timesheet.CalculatePay( Wage );
   }
}
```

PayrolFixtures.WeeklyCompensation			
Standard Hours	Holiday Hours	Hourly Wage	Pay( )
40	0	10	$400
40	0	20	$800
41	0	20	$830
40	1	20	$840
41	1	20	$870

Walkthrough: Column Fixture Test

```
public class WeeklyCompensation ColumnFixture {

    public int StandardHours;

    public int HolidayHours;

    public Currency Wage;

    public Currency Pay( ) {
        WeeklyTimesheet timesheet = new
            WeeklyTimesheet( StandardHours, HolidayHours);
        return timesheet.CalculatePay( Wage );
    }
}
```

PayrollNormalWeeklyCompensation			
StandardHours	HolidayHours	HourlyWage	Pay()
40	0	10	$400
40	0	20	$800
41	0	20	$830
40	1	20	$840 *expected* $800 *actual*
40	1	20	$870 *expected* $830 *actual*

Other Kinds of Fit Fixtures

- **Row Fixtures are used to specify expected contents of tables**

PayrolFixtures.EmployeeCompensationThisWeek			
Number	Name	Hours	Pay
10001	Gerard Meszaros	45	$950
10002	John Smith	40	$800
10003	Jane Doe	42 *expected* 40 *actual*	$860 *expected* $800 *actual*
10999 *surplus*	No Longer Here	20	$800

- **Action Fixtures and Do Fixtures are used to specify workflows**

Details of these are beyond the scope of this tutorial

Sample Multi-fixture Test

- **Load data into table using Column Fixture:**

PayrolFixtures.EmployeeTableLoad			
Number	Name	Hours	Add()
10001	Gerard Meszaros	45	OK
10002	John Smith	40	OK
10003	Jane Doe	420	OK *expected* Too Big *actual*

- **Exercise system using Action Fixture**

PayrolFixtures.RunPayrollActionFixture			
RunJob	Payroll		

- **Verify database contents using Row Fixture**

PayrolFixtures.EmployeeCompensationThisWeek			
Number	Name	Hours	Pay
10001	Gerard Meszaros	45	$950
10002	John Smith	42 *expected* 40 *actual*	$860 *expected* $800 *actual*
10003 *surplus*	Jane Doe	20	$800

Exercise: Fit Testing

- **You are building a system with complex calculations. You wish to capture these calculations as an executable specification using the Fit Framework.**
- **Translate (the small subset of) the business rules described next into a set of inputs with their corresponding outputs using the Fit Table format supplied**

Fit Exercise: Business Rules

- "Finished" goods get taxed at 5% local tax and 8% Federal tax, unless they cost less than 1.00 in which case the local tax is waived
- "Raw" goods get taxed at 10% local and 8% federal, but the Federal rate decreases to 6% when the product costs less than $2000 and 5% when less than $1000.
- "Food" gets taxed at 5% federal regardless of price

Use the following Column Fixture to prepare automated Fit tests for the TaxCalculator:

Fit Exercise: Worksheet

Fit.TaxCalculationTester				
Type	Price	Local Tax( )	Federal Tax( )	Total Cost( )

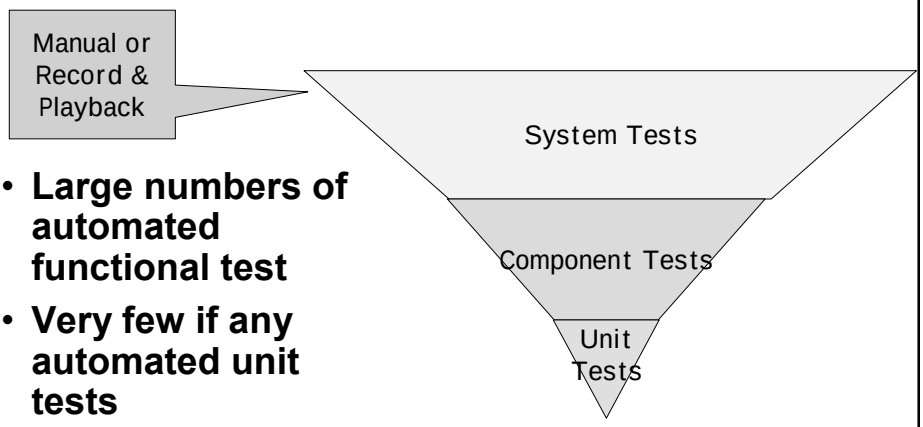
Fit Exercise: Worksheet

Fit.TaxCalculationTester				
Type	**Price**	**Local Tax( )**	**Federal Tax( )**	**Total Cost( )**

Applicability of Fit Tests

- **Because Fit is software, it can be used for:**
  - Component Testing
  - System/Application Testing
- **Most appropriate for testing business logic**
- **Often used for testing business workflow**
- **Can be used for testing User Interfaces**
  - But not really designed for this
- **Fit encourages Design for Testability**
- **Fit reduces the amount of manual testing required**
  - By changing focus to UI rather than business logic

Test Automation Pyramid

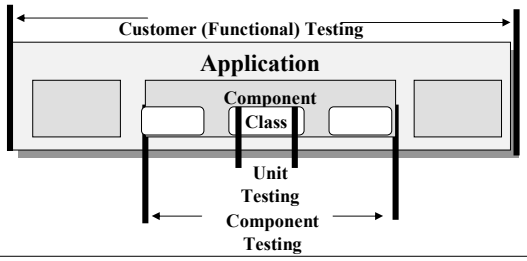


- **Large numbers of automated functional test**
- **Very few if any automated unit tests**

Typical when testing (& automation) is "QA's job"

Why We Need Unit/Component Tests

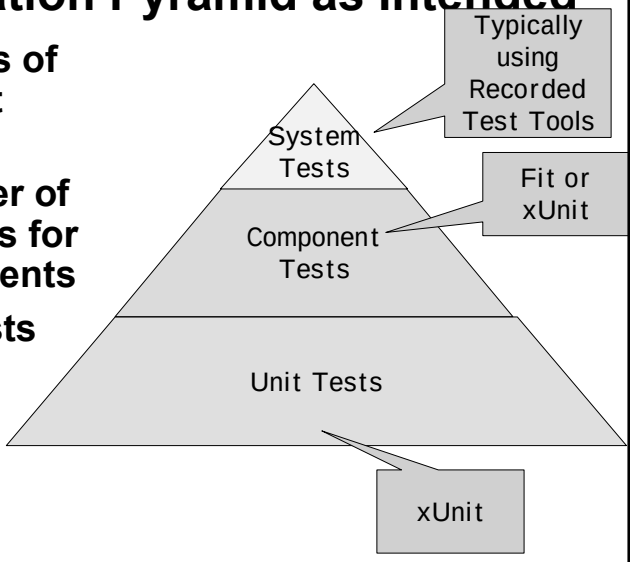
- **Functional Tests will tell us which features of the product doesn't work**
- **Component tests will tell us which component is at fault**
- **Unit tests tell us exactly which class/method is broken**



Unit Tests also let us test code we cannot hit in functional tests

Test Automation Pyramid as Intended

- **Large numbers of very small unit tests**
- **Smaller number of functional tests for major components**
- **Even fewer tests for the entire application & workflow**



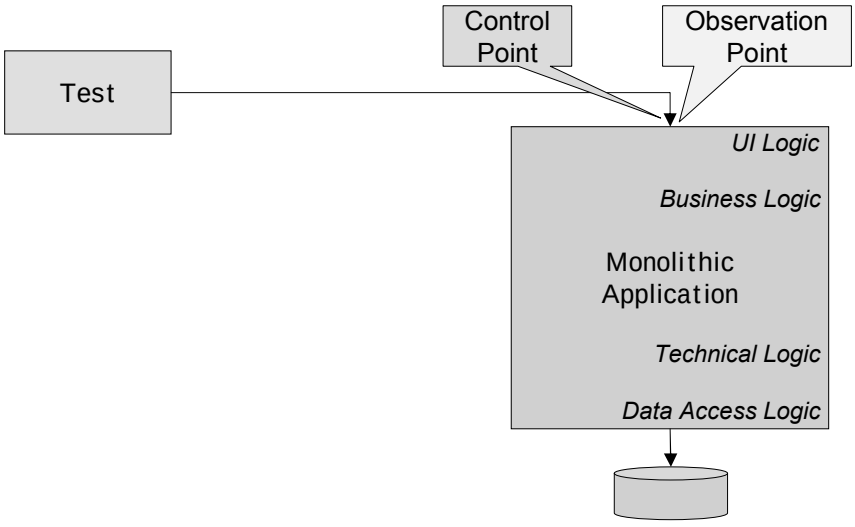
What Does It Take to Do This?

- **Communication & Collaboration**
  - To know what was well tested at Unit / Component level
  - To trust those tests enough to not repeat at Functional Test level
- **Design For Testability**
  - To make testing at Unit / Component level easy enough

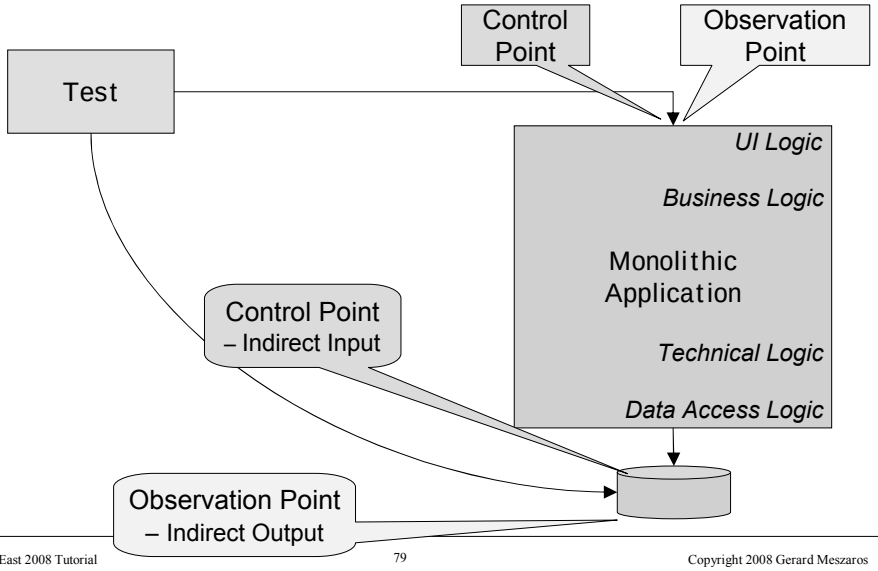
Design for Testability

- **Layers**
- **Control Points**
  - Direct Inputs
  - Indirect Inputs
- **Observation Points**
  - Direct Outputs
  - Indirect Outputs
- **Design for Substitutability**

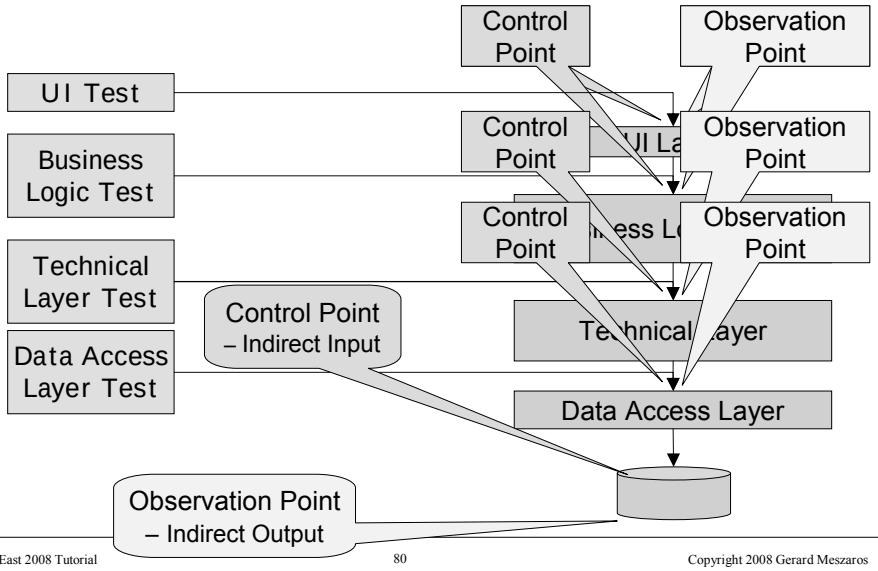
Application *not* Designed for Testability



Testing Database-Centric Application



Application Designed for Testability



Layer Testing - On-Stack

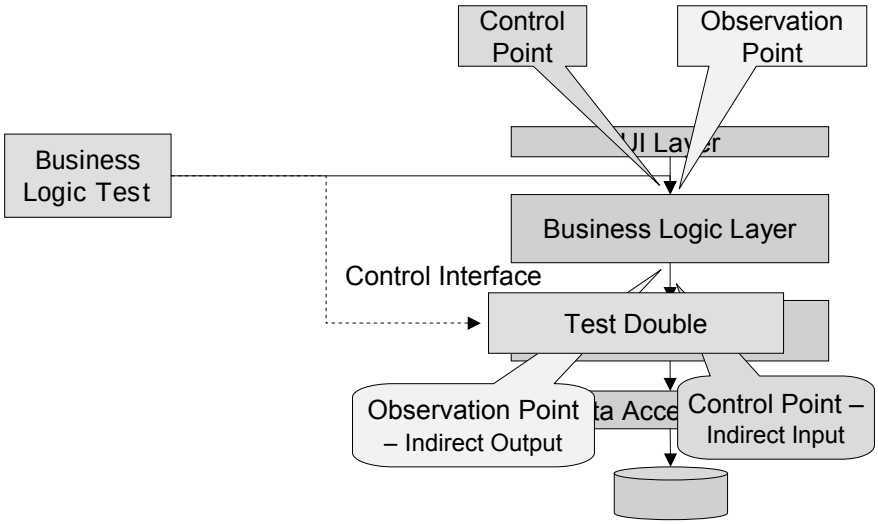
Advantages

- Can test without client
- Helps identify where the defect occurs
  - » **Based on which layers failing tests are in**

Issues

- Need tests that characterize each layer or component
  - » **Must accurately represent how client use it**
- Need all dependencies before we can test

Application Designed for Substitutability



Collaborative Approach

- **Testing cannot make this happen on their own**
- **Development**
  - often isn't motivated to do it for themselves
  - often doesn't have the testing experience to do it
- **Together, Testing & Development can make it happen**

Benefits of Collaboration

- **By helping development test, Developers become interested in:**
  - Test automation
  - Design-for-Testability
- **Both of these making Testing's job a lot easier**
  - Access to development expertise
  - More testable designs
  - Lower cost test automation
- **Developers Lives Improve**
  - New challenges & skills
  - Better quality results

Agenda

- **The State of the Union**
- **S/W Development Value Stream**
- **How Can Testers Help Development?**
- **How can Testers help Product Management?**
  - Define product concept
  - Validate product design
  - Clarify requirements
- **Collaboration Techniques**
- **Summary & Closing Comments**

Why Help Product Management?

- **Better Quality Product**
  - Better fitness for purpose by:
    - » **understanding users better**
    - » **designing product for actual users**
- **More Job Satisfaction**
  - Having a *Real* impact on quality
  - Represent end-users better

How to Help Product Management?

- **Represent End Users in Envisioning Product**
  - Product Vision Workshops
  - Paper Prototyping of Envisioned Product
- **Involve End Users in Validating Product Concept**
  - Usability Testing of Paper Prototypes

Product Envisioning

- **Multi-disciplinary workshops for envisioning product to be built**
  - Get everyone's inputs
  - Get everyone's buy-in
- **Several potential outputs:**
  - Elevator Statement
  - "Product Box"
  - Paper Prototype

Elevator Statement

- For (target customers - beachhead segment only)
- Who are dissatisfied with (the current market alternatives)
- Our product is a (new product category)
- That provides (key problem-solving capability)
- Unlike (the product alternative)
- We have assembled (key "whole product" features for our specific application)

Crossing the Chasm - Marketing and Selling High-Tech Products to Mainstream Customers.
By Geoffrey A. Moore - See P154

Product Box

- Product Graphic
- 15-20 main features
- 3-4 key differentiating features

- Helps team focus on what's really important

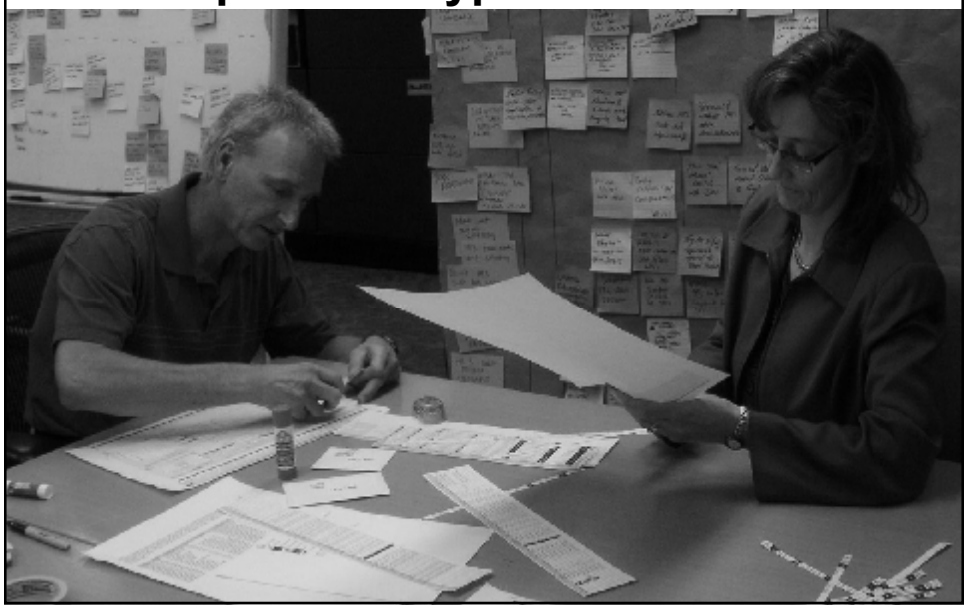
Paper Prototyping

Shared Vision

- **Goals:**
  - Turn nebulous product ideas into something tangible
  - Get feedback on proposed designed early & cheaply
  - Avoid unnecessary rework of design after code is built
- **Method:**
  - Define screens to accomplish specific task
  - Use paper, pens, tape, scissors, etc. to mock up
  - Can use a mix of hand-drawn & software generated elements

Warning: The more finished it looks, the less feedback you'll get!

Paper Prototypers in Action!



Benefits of Paper Prototyping

Testers

- Understand the user better
- Understand the rationale behind the product
- Understand the product much earlier
- Influence the product design while it is still easy to influence

Project

- Get input from testers perspective while it can influence the design
- Arrive at a better design
- Get tests informed & involved in usability testing

Sample UxD Design Process

- Identify your users' needs
- Identify the workflow
- Identify the screens & tasks
- Understand the tasks to be supported
  - Identify information the user needs
  - Identify information the user needs to supply
  - Identify the actions the user will ask the computer to do
- Propose one (or more) candidate design
- Test the design with real users

Identify Your Users' Needs

- **What task do they need help in doing**
- **What are the characteristics of the users themselves**
  - Experienced computer users?
  - Computer novices?
- **What are the characteristics of their usage?**
  - Will they be using the software regularly?
  - Or very occasionally?
  - Will they be trained in it's usage?
  - Or expected to learn it alone?

Building the Paper Prototype

- **For each Screen needed for a task**
  - Use a sheet of paper for the window
  - Add a title bar
  - Add fields & labels and buttons or links
  - Add any frames needed to identify groupings of fields & buttons
- **Build all widgets on Post-it™ notes for ease of refactoring the design**
- **Next, test your design!**

Exercise: Paper Prototyping

- **Application: Online feeding regimen analysis**
- **User: Hobby farmer with several horses and rudimentary computer skills**
- **Task: Get analysis of current diet of one horse**

Key Features:

Data Entry Features

- **Horse profile (breed, weight, age, activity level, metabolism, current condition)**
- **Current & Available feeds:**
  - Choose one or more feeds plus amount per day
  - "Pick lists" of common commercial feeds
- **Selectable goal (gain, lose, maintain weight)**

Reporting Features:

- **Text report**
- **Graphical report**
- **Printing**

Misc Features:

- **Add custom feed to feed database**

Exercise: Paper Prototyping

- **Application: Appliance Part Finder**
- **User: DIY home owner with a broken appliance**
- **Task: Get a price/availability on a part**

Key Features:

Part Specification:

- **Type of appliance**
  - {Stove, Fridge, Dishwasher, Toaster, etc.}
- **Part Description**
  - Free text
- **List of keywords**
  - Predefined for each type of appliance
- **Either Description or at least one Keyword must be supplied**

Search Results:

- **List of possible parts**
  - Name, thumbnail, part#
- **A way to fetch prices at online sellers**
  - For selected part(s)
- **A way to find local dealers**
  - For selected part(s)

Agenda

- **The State of the Union**
- **S/W Development Value Stream**
- **How Can Testers Help Development?**
- **How can Testers help Product Management?**
  - Define product concept
  - Validate product design
  - Clarify requirements
- **Collaboration Techniques**
- **Summary & Closing Comments**

Wizard of Oz Testing

- **Purpose:**
  - Find design defects quickly
  - Make design decisions based on data, not speculation
- **How:**
  - Test early versions of the design with users
  - Gather data on alternative design options

Wizard of Oz Testing

- **Roles:**
  - 1 Test Facilitator (business SME)
  - 2 people playing computer, coprocessor & HELP system (anyone)
  - 2-3 observers (developers + business)
  - 1 or 2 test subjects
- **Preparation:**
  - Task descriptions – what the user will attempt to do
  - Paper Prototypes – what the user will use to do it
- **Test Sessions:**
  - Typically about 1 hour each
  - Tested with pairs of end users (co-discovery)

Wizard of Oz Testing

Test Session Workflow

1. **Facilitator describes testing process**
   1. How the user and computer will interact
      - **By pointing or writing**
   2. The role of the observers
      - **"Please speak your thoughts so they can record them"**
   3. How the user can ask for help
      - **"Point to the HELP button in the top right corner"**
2. **Facilitator provides user(s) with task to do**
3. **User attempts to do task using "application"**
   - By "clicking" and "typing" in "application"
4. **Observers record any problems encountered**
5. **Facilitator debriefs the user**
   - Was there anything that particularly confused you?

User-Computer Interaction

1. **Computer lays down start screen**
   - Optionally, Computer places a sheet of plastic film over "screen" to simplify erasing user inputs
2. **User interacts with screen**
   - Points at button or link and says "I'm clicking on this .."
   - Writes text into an input field with (erasable) pen(cil)
3. **Facilitator may ask for clarification of what the user is doing**
   - User should not be allowed to ask questions, but can click on Help icon
   - Help system may respond verbally with helpful message/hints
4. **Computer responds to user inputs**
   - By laying down next screen over first
   - By laying down pop-up dialog boxes or pick lists
   - Co-processor (computer helper) may pre-populate next screen with relevant data
5. **Facilitator debriefs the user**
   - Was there anything that particularly confused you?

What Observers Record

- User seemed confused when …
- User could not find a way to …
- User didn't notice the …; they seemed to be looking for <describe what they thought it should be>
- User didn't like <describe UI device>
- User said they hate/love a specific aspect of the UI

Benefits of Wizard of Oz Testing

Testers

- Understand the user better
- Understand the rationale behind the product
- Understand the product much earlier
- Influence the product design while it is still easy to influence
- Become more valuable to the team

Project

- Earlier feedback from users
- Get testers more familiar with users and product
- Bring testing mindset earlier in project
- Avoid late-breaking requirements (bugs)

Exercise: Wizard of Oz Testing

- **Using the materials provided, test the paper prototype for usability**

Wizard of Oz Test Observation Sheet

Task: _____

Screen	Location on Screen	What the user was doing/trying	What happened? What user said.

Wizard of Oz Test Observation Sheet

Task: _____

Screen	Location on Screen	What the user was doing/trying	What happened? What user said.

Benefits

- Testers get better understanding of application
- They get it much earlier in project
- Better understanding of users
- Better relationship with Product Management
- More valuable to organization
- Less likely to be offshored

Agenda

- **The State of the Union**
- **S/W Development Value Stream**
- **How Can Testers Help Development?**
- **How can Testers help Product Management?**
  - Define product concept
  - Validate product design
  - Clarify requirements
- **Collaboration Techniques**
- **Summary & Closing Comments**

Clarifying Requirements

- **Show what "done looks like"**
  - Stephen Covey's 2nd "Habit": "**Begin with the End In Mind"**
- **Enumerate all the capabilities that must be supported**
  - Identify the success paths
  - Identify all the failure & error scenarios
- **Provide detailed examples of input data and responses**

"*The Seven Habits of Highly Effective People*"
by Stephen R. Covey published by Free Press

Clarifying Requirements

All the benefits that Development gets from understanding what they build …

… apply equally to Product Management understanding what needs to be built

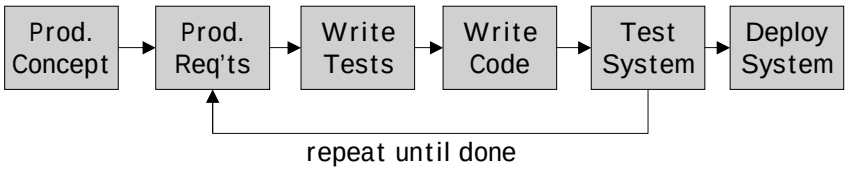
- More complete requirements
- More usable product
- Better estimates

- **Techniques like**
  - Identifying failure scenarios
  - Fit tests for business rules and workflows
  - etc.

Refer back to previous section

Summary of Ways to Help

- **Help Product Management determine product concept**
- **Help Product Management flesh out requirements**
- **Help Development understand "What done looks like"**
- **Give Development timely feedback on how it's doing**
- **Give Product Management an accurate scorecard of how the project is doing**



Agenda

- The State of the Union
- S/W Development Value Stream
- How Can Testers Help Development?
- How can Testers help Product Management?
- Collaboration Techniques
- Summary & Closing Comments

What is Collaboration?

Dictionary.com Definition:

– Collaborate: To work together, especially in a joint intellectual effort

Our Definition:

– A team of people working together to solve a shared goal

What isn't Collaboration?

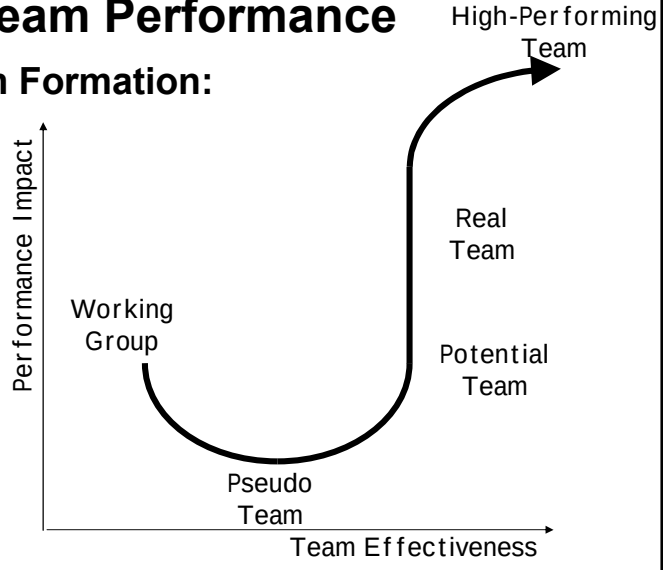
- **Command and Control**
  - Knowledgeable leader derives solution and directs team members to implement subcomponents.
  - Loss of leader results in chaos.
- **Competence based leadership**
  - Recruiting the best & the brightest
  - Encourage them to excel individually.
  - Loss of individual results in significant loss of knowledge.
- **Cultivating Cultures**
  - Cultivation of individuals to excel independently
  - Knowledge is prized over deliverables.
  - Teamwork is not encourage

None of these harness the power of the team

Team Performance

Phase of Team Formation:

- **Forming**
- **Storming**
- **Norming**
- **Performing**

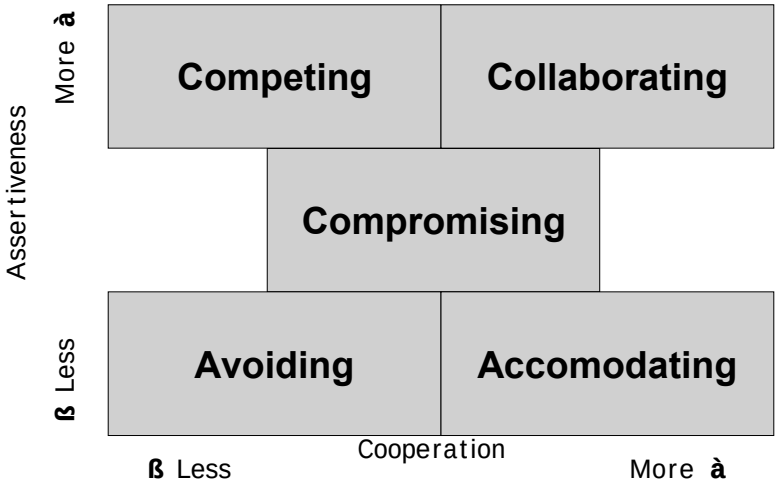


Team Performance Curve by Jon Katzenback and Douglas Smith)

Collaborative Team Characteristics

- Self-Organizing
- Empowered to make decisions
- Belief in Vision & Success
- Committed to Success
- Trust Each Other
- Participatory Decision Making
- Consensus-Driven
- Constructive Disagreement

Team Conflict Dynamics



Thomas-Kilmann conflict mode instrument (TKI)

What is Concensus?

Not:

- "Wild agreement between all team members"

But:

- "I can live with that and support it."

No one:

- disagrees vehemently with decision
- had to *compromise* their principles
- is agreeing just to *accommodate* the group

Consensus requires individual discipline

Collaboration Techniques

- **Pairing on Tasks**
- **Joint Deliverables**
- **Collaborative Team Events**

Pairing on Tasks

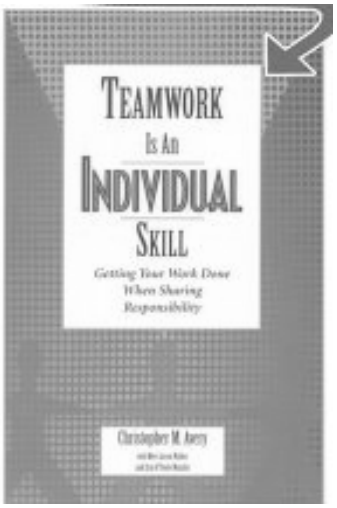
- **Many opportunities to help someone with their tasks**
  - Two developers writing code and unit tests together
  - Developer help a tester automate some functional tests
  - Tester help developer unit test some code
  - Tester help analyst with identifying missing cases in requirements, etc.
  - Project manager and Development Lead or Test Lead bouncing ideas off each other
  - A mentor helping someone through a problem

Joint Deliverables

- **Producing a deliverable that requires two different skills to produce.**
- **Example: Fit Tests**
  - Developers know how to code Fit fixtures
  - Analysts (or business people) can produce examples of requirements
  - Working together, they can produce a executible example (aka execution specification)

Teamwork is an Individual Skill

- **Getting Your Work Done When Sharing Responsibilities**
- **Christopher W. Avery**
- **ISBN: 1576751554**
- **Techniques for how to be a good collaborative team member**



Collaborative Team Events

Two main kinds of activities:

1. Data gathering or generation
2. Decision Making

Events with larger numbers of participants require strong facilitation

- More Structure
  - » **Prepare**
  - » **Prompt**
  - » **Gather**
  - » **Process**
- Ground rules (owned by the team)
- Breakout sessions with “Report Backs”

Data Gathering Techniques

Brainstorming
- **Facilitator-led Callout**
- **Post-it Notes**
- **Round Robin**
- **Pass the Pen**
- **Pass the Card**

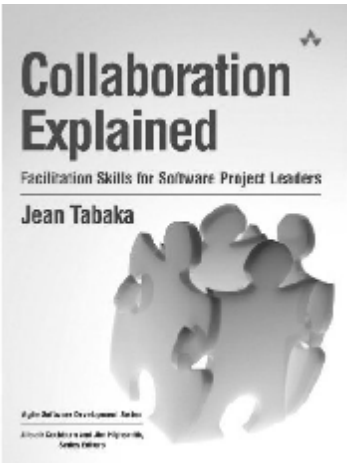
Listing
- **Facilitator-led Listing**
- **Post-it Notes Listing**
- **Round Robin Listing**

Decision Making Techniques

- **Structured Discussion**
  - Agenda designed to flow from gathering information to making a consensus decision
    - » **What are all the possible artifacts we could produce?**
    - » **Based on past experience, what benefits do they provide?**
    - » **What are the downsides to these artifacts?**
    - » **What other considerations should affect our choice?**
    - » **What is the final set of artifacts we will use for this project?**
- **Prioritization**
  - Dot Voting, etc.
- **Parking Lots**
  - To avoid distracting discussions

Collaboration Explained

- Facilitation Skills for Software Project Leaders
- By Jean Tabaka
  - ISBN: 0321268776
- Excellent resource for facilitating agile project meetings



Encouraging the Right Behavior

See cartoon at:
- **http://i18.tinypic.com/34461wi.jpg**

or:
- **http://www.flubu.com/comics/dilbert2.gif**

Or, search Google Images for
'dilbert "write me a minivan"'

- Collaboration won't happen if individualism is what is rewarded.
- Whole Team needs to sink/swim together
- Reward system needs to reinforce this
- e.g Team Performance Ranking

Encouraging Collaboration

- Foster Self-Organization
- Stay Positive
- Ask Questions
- Encourage Information Sharing
- Drive to Concensus
- Make Everything Highly Visible
- Take Away the Blame

Agenda

- The State of the Union
- S/W Development Value Stream
- How Can Testers Help Development?
- How can Testers help Product Management?
- Collaboration Techniques
- Summary & Closing Comments
  - Summary of Key Points
  - References
  - Q & A
  - Sample Exercise Solutions

Summary of Key Points

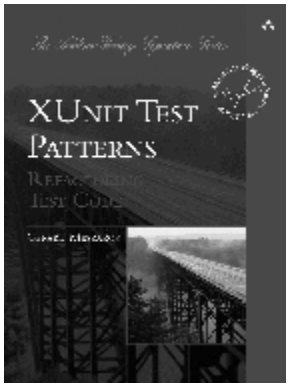
- **Testers can add value well beyond testing**
- **Testing needs to help other roles avoid waste**
- **Testers can help:**
  - Define the product concept
  - Validate the product design
  - Clarify requirements
  - Help development avoid defects
  - Provide development with early & timely feedback
- **Lean Principles can help guide improvements**
  - Build Quality In
  - Optimize the Whole
  - Avoid Waste

Useful References

- **Any book on Teamwork**
- **Collaboration Explained**
  - Facilitation Skills for Software Project Leaders
  - By Jean Tabaka ISBN: 0321268776
- **Teamwork is an Individual Skill**
  - Getting Your Work Done When Sharing Responsibilities
  - Christopher W. Avery ISBN: 1576751554
- **Fit for Developing Software**
  - Rick Mugridge & Ward Cunningham
  - ISBN:

Thank You!

Gerard Meszaros
StarEast2008@gerardm.com
website:
http://www.xunitpatterns.com
http://www.gerardmeszaros.com



Jolt Productivity Award winner - Technical Books

Call me when you:

- **Need help with test automation strategy**
- **Want to remove waste from your process**
- **Want to teach developers how to unit test**
- **Want to transition to agile or lean**
- **Want to do agile/lean better**
- **Want to improve usability of your applications**

Exercise Solutions

Fit Exercise Solution – Basic

Fit.TaxCalculationTester				
Type	Price	Local Tax( )	Federal Tax( )	Total Cost( )
Finished	0.99	0.00	0.08	1.07
Finished	1.00	0.05	0.08	1.13
Raw	999.99	100.00	50.00	1,149.99
Raw	1000	100.00	60.00	1,160.00
Raw	1,999.99	200.00	120.00	2,319.99
Raw	2000	200.00	160.00	2,360.00
Food	10.00	0.00	0.50	10.50

This is simply a set of executable examples or tests. The business rules need to be reverse-engineered from these examples. The next page shows how to document the business explicitly in the same document.

Fit Exercise Solution - Better

"Finished" goods get taxed at 5% local tax and 8% Federal tax, unless they cost less than 1.00 in which case the local tax is waived. "Food" gets taxed at 5% federal regardless of price.

Fit.TaxCalculationTester				
Type	Price	Local Tax( )	Federal Tax( )	Total Cost( )
Finished	0.99	0.00	0.08	1.07
Finished	1.00	0.05	0.08	1.13
Food	10.00	0.00	0.50	10.50

"Raw" goods get taxed at 10% local and 8% federal, but the Federal rate decreases to 6% when the product costs less than $2000 and 5% when less than $1000.

Fit.TaxCalculationTester				
Type	Price	Local Tax( )	Federal Tax( )	Total Cost( )
Raw	999.99	100.00	50.00	1,149.99
Raw	1000	100.00	60.00	1,160.00
Raw	1,999.99	200.00	120.00	2,319.99
Raw	2000	200.00	160.00	2,360.00

Fit Exercise Solution – Better (P2)

"Finished" goods get taxed at 5% local tax and 8% Federal tax, unless they cost less than 1.00 in which case the local tax is waived

Fit.TaxCalculationTester				
Type	Price	Local Tax( )	Federal Tax( )	Total Cost( )
Raw	999.99	100.00	50.00	1,149.99
Raw	1000	100.00	60.00	1,180.00
Raw	1,999.99	200.00	120.00	2,319.99
Raw	2000	200.00	160.00	2,360.00

